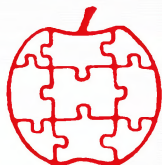


Apple

\$1.80



Assembly

Line

Volume 6 -- Issue 8

May, 1986

In This Issue...

DOS 3.3 for the UniDisk 3.5.	2
Recovering & Repairing Lost Programs	18
More and Better Division by Seven.	20

Enhancing Applesoft with the Toolbox Series

A number of years ago, when Roger Wagner Publishing was still called Southwestern Data Systems, he published Peter Meyer's program "The Routine Machine". The system evolved into four packages: Wizard's Toolbox, Database Toolbox, Video Toolbox, and Chart'n Graph Toolbox. Each "Toolbox" contains a large assortment of assembly language routines which enhance the capabilities of Applesoft. The "Workbench" (included with each Toolbox) allows programmers to add any assortment of these routines to their Applesoft programs at any time. The routines are all called by using the ampersand (&) statement.

Roger will make a special deal for Apple Assembly Line subscribers: he'll send a free copy of the "Trial-size Toolbox" (normally \$3) to anyone who mentions reading about the package here. The disk includes eight ampersand commands, including a charting command-set with 12 sub-commands, a fixed-length input command, and a print with word-wrap command. All are usable under either DOS 3.3 or ProDOS. Also on the disk is the text of a 50-page manual. The manual includes a tutorial for the toolbox system, a complete explanation of the commands included on the sampler disk, and a comprehensive listing of every command in each of our Toolbox packages. For the free sampler write to Roger Wagner Publishing, Box 582, Santee, CA 92071.

The Toolbox packages are normally \$39.95 each. We'll sell them here at S-C for \$36 each, or \$140 for the complete set.

DOS 3.3 for the UniDisk 3.5 (RWTS 3.5).....Bill Morgan

We finally got one of Apple's new UniDisk 3.5 drives for the //e, and let me tell you it's very nice. This small but large addition to our favorite computer is about half the volume of a Disk II, but each disk stores almost six times as much information. It's even a bit faster than the 5.25" drives, about 1.3 times the speed.

Of course there's a catch. In line with Apple's policy of supporting ProDOS only, the new device doesn't use DOS 3.3, at least not as far as Apple is concerned. There are already several different UniDisk versions of DOS, and we're about to build our own right here. It's really quite easy.

There are two parts to the problem: intercepting and handling RWTS calls to the UniDisk slot, and formatting a 3.5" disk with a DOS VTOC and Catalog.

There are a variety of ways to take over a call to RWTS. When we call RWTS at \$3D9 it jumps on to \$B7B5, where interrupts are disabled before calling the real RWTS entry at \$BD00. Some programs take control at \$B7B7 and others at \$BD00. I looked at the code at \$BD00 and saw that it does a little housekeeping and then at \$BD10 loads the accumulator with the slot*16 value from the IOB. That looks like the ideal time to check to see if this call is for my slot, so \$BD12 is where I patch in the jump to my code. If you are using several nonstandard devices with DOS 3.3 (Sider or other hard disk, RAM disk, other drives) you will need to keep track of who's patching into RWTS where.

Now we come to the question of where to put our version of RWTS. There's certainly no room inside DOS for almost a page of code plus two pages of buffer. I thought I could probably squeeze the code into page three, but that still left that buffer (not to mention the crowd already living at that popular address!) It occurred to me to throw INIT away and put the code inside the existing RWTS at \$BEAF, but what about the buffer? I finally decided to use the time-honored technique of moving the DOS buffers and HIMEM down and installing my program and buffer in there. That's also crowded, but where isn't? The first working version of RWTS 3.5 ran at \$9900, with the buffer at \$9B00-9CFF. The installation routine checked to see if anyone else was using the space and returned an error if so. Applesoft and the S-C Macro Assembler got along with this arrangement just fine, so I spent some time polishing the program and started to write this article.

That's when I was forcibly reminded that the S-C Word Processor sets its own HIMEM and is firmly convinced that \$9900-99FF is the buffer for characters deleted off the screen. In other words, the first time I tried to save some text to the UniDisk it blew sky high. I had decided to live without the Word Processor on the UniDisk for the time being when I noticed a couple of interesting things in Beneath Apple DOS. There is a 342-byte buffer inside RWTS at \$BB00-BC55, and the code immediately after that buffer is called only by INIT! There really are two full pages of available buffer space inside DOS

S-C Macro Assembler Version 2.0.....DOS \$100, ProDOS \$100, both for \$120
Version 2.0 DOS Upgrade Kit for 1.0/1.1/1.2 owners.....\$20
ProDOS Upgrade Kit for Version 2.0 DOS owners.....\$30
Source Code of S-C Macro 2.0 (DOS only).....additional \$100
Full Screen Editor for S-C Macro (with complete source code).....\$49
S-C Cross Reference Utility.....without source code \$20, with source \$50
RAK-Ware DISASM.....without source code \$30, with source \$50
S-C Word Processor (with complete source code).....\$50
DPI8 Source and Object.....\$50
Double Precision Floating Point for Applesoft (with source code).....\$50
ES-CAPE (Extended S-C Applesoft Program Editor).....
Including Version 2.0 With Source Code.....\$50
ES-CAPE Version 2.0 and Source Code Update (for Registered Owners)....\$30
"Bag of Tricks 2".....(\$49.95) \$45 *
Applesoft Toolbox Series, Roger Wagner Publishing.....each (\$39.95) \$36 *
all four (\$159.80) \$140 *
MacASM -- Macro Assembler for Macintosh (Mainstay).....(\$150.00) \$140 *
S-C Documentor (complete commented source code of Applesoft ROMs).....\$50
Source Code of //e CX & F8 ROMs on disk.....\$15
Cross Assemblers for owners of S-C Macro Assembler....\$32.50 to \$50 each
(Available: 6800/1/2, 6301, 6805, 6809, 68000, Z-80, Z-8, 8048,
8051, 8085, 1802/4/5, PDP-11, GIL650/70, others)
AAL Quarterly Disks.....each \$15, or any four for \$45
Each disk contains the source code from three issues of AAL,
saving you lots of typing and testing.
The quarters are Jan-Mar, Apr-Jun, Jul-Sep, and Oct-Dec.
(All source code is formatted for S-C Macro Assembler. Other assemblers
require some effort to convert file type and edit directives.)
Diskettes (with hub rings)..... package of 20 for \$20 *
Vinyl disk pages, 6"x8.5", hold two disks each.....10 for \$6 *
Diskette Mailing Protectors (hold 1 or 2 disks).....40 cents each
(Cardboard folders designed to fit 6"x9" Envelopes.) or \$25 per 100 *
Envelopes for Diskette Mailers..... 6 cents each
65802 Microprocessor (Western Design Center).....(\$95) \$50 *
quickLoader EPROM System (SCRG).....(\$179) \$170 *
PROMGRAMMER (SCRG).....(\$149.50) \$140 *
Switch-a-Slot (SCRG).....(\$179.50) \$170 *
"65816/65802 Assembly Language Programming", Fischer.....(\$19.95) \$18 *
"Programming the 65816", Eyes.....(\$22.95) \$21 *
"Apple //e Reference Manual", Apple Computer.....(\$24.95) \$23 *
"Apple //c Reference Manual", Apple Computer.....(\$24.95) \$23 *
"ProDOS Technical Reference Manual", Apple Computer.....(\$29.95) \$27 *
"Now That You Know Apple Assembly Language...", Gilder.....(\$19.95) \$18 *
"Apple ProDOS: Advanced Features for Programmers", Little..(\$17.95) \$17 *
"Inside the Apple //c", Little.....(\$19.95) \$18 *
"Inside the Apple //e", Little.....(\$19.95) \$18 *
"Apple II+/IIe Troubleshooting & Repair Guide", Brenner.....(\$19.95) \$18 *
"Apple II Circuit Description", Gayler.....(\$22.95) \$21 *
"Understanding the Apple II", Sather.....(\$22.95) \$21 *
"Understanding the Apple //e", Sather.....(\$24.95) \$23 *
"Enhancing Your Apple II, vol. 1", Lancaster.....(\$15.95) \$15 *
"Enhancing Your Apple II, vol. 2", Lancaster.....(\$17.95) \$17 *
"Assembly Cookbook for the Apple II/IIe", Lancaster.....(\$21.95) \$20 *
"Beneath Apple DOS", Worth & Lechner.....(\$19.95) \$18 *
"Beneath Apple ProDOS", Worth & Lechner.....(\$19.95) \$18 *
"Real Time Programming -- Neglected Topics", Foster.....(\$9.95) \$9 *
"Microcomputer Graphics", Myers.....(\$14.95) \$14 *
"Assem. Language for Applesoft Programmers", Finley & Myers.(\$18.95) \$18 *
"Assembly Lines -- the Book", Wagner.....(\$19.95) \$18 *
"AppleVisions", Bishop & Grossberger.....(\$39.95) \$36 *

* On these items add \$2.00 for the first item and
\$.75 for each additional item for US shipping.

Foreign customers inquire for postage needed.

Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***

*** (214) 324-2050 ***

*** Master Card, VISA, Discover and American Express ***

along with room for the code.

So this edition of RWTS 3.5 runs at \$BEAF, with its buffer at \$BB00-BCFF. I did hit one more snag when I went to use that buffer area; \$BCDF-BCFF is officially unused, which means it's a popular place for other patches. My system has part of our fast LOAD/BLOAD patch (AAL April 83) there, so I had to shave a few more bytes out of my program to make room to move the LOAD patch up to \$BF97-BFB7. You may have to make some such adjustment, so be sure to check for some other patch at \$BCDF.

The UniDisk 3.5 uses a new software interface, called the Protocol Converter. The PC is a sort of serial bus, which can have several devices daisy-chained to the same controller. We program the PC with a calling structure very similar to the ProDOS MLI calls. Here's an example:

```
CALL JSR DISPATCH
      .DA #1      read command
      .DA PARMLIST
      BCS ERROR
      ... whatever code

PARMLIST
      .DA #3      3 parameters
      .DA #1      unit number
      .DA BUFFER  buffer address
      .DA <BLOCK  block number (3 bytes)
```

That's all it takes to read a 512-byte block into our buffer. Notice that this standard specifies a 3-byte block number: all current devices use only two bytes of the block number, but they're allowing for expansion beyond 32 megabytes. The unit number isn't the same as a ProDOS unit; this is the position of the device in the PC chain. We need to look up the value of DISPATCH in the card. The byte at \$CsFF (s = slot) contains the offset into the ROM of the ProDOS driver entry and the Protocol Converter entry is defined to be 3 bytes after that. For example, in my UniDisk 3.5 controller in slot 5 the byte at \$C5FF is \$0A. That means that the ProDOS entry to the card is \$C50A and the PC entry is \$C50D.

There's a quick look at the Protocol Converter. We haven't seen much information published about it yet. The new //c Technical Reference Manual has a good section, including a ROM listing, but the //e UniDisk 3.5 includes no programmer's documentation. Bob is planning a more extensive article on its programming for next month's AAL. Stay tuned...

Apple's new memory expansion card has a PC interface and this RWTS will work with that card as well, but some modification will be needed to use more than one PC at a time. The installation code could scan all slots looking for PCs and build a table of valid slots and entry addresses. Then the initial code at MY.RWTS could search that table and plug the appropriate PC.DISPATCH address into the calls.

The Protocol Converter sees the UniDisk as 1600 blocks of 512

bytes each, for a total of 819,200 (800K) bytes of storage. We have no way to find out about actual tracks and sectors on the disk; this drive seems to use the Macintosh scheme of a variable number of blocks per track. Therefore, we're going to translate DOS's tracks and sectors into some block number and ask the PC for that block, not worrying about where it actually comes from.

The VTOC on a DOS disk has room for 50 tracks of 32 sectors each. That adds up to 400K, or exactly half a UniDisk, so we should be able to set things up with 2 logical drives of 400K each. The number of tracks per disk and the number of sectors per track are both stored as parameters in the VTOC as well, just to make things easier. Two drives per disk means that we can put drive one in the lower 800 blocks and drive two in the upper 800. Figuring that 32 sectors per track means 16 blocks per track and two sectors per block gives us this equation:

$$\text{BLOCK} = (\text{DRIVE}-1)*800 + \text{TRACK}*16 + \text{SECTOR}/2$$

An even-numbered sector is in the lower half of a block, odd in the upper half.

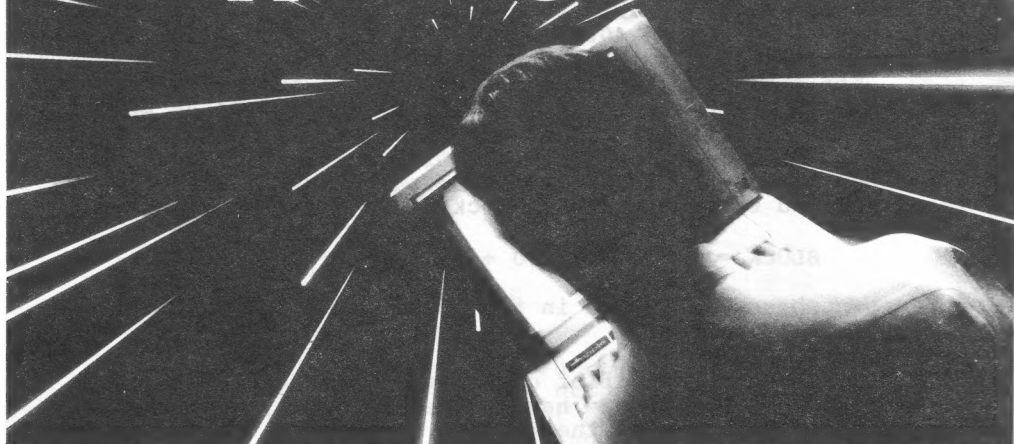
Since each sector is half of one block on the disk, we can't just write one sector. We have to read a block, copy the new information into half of the buffer, then write that block back out. This takes extra time, but simplifies some of the control logic because every call does a read first.

That first working version of RWTS 3.5 did a new read for every read call, and a new read and write for every write. Well that proved to be much too slow, even slower than the old Disk II. Then I realized that nearly all DOS operations are reading or writing consecutive sectors in a file, so I must be spending a lot of time reading a block that was already in my buffer just to get the sector in the other half of the block. Sure enough, the performance almost doubled when I started keeping track of which block was in the buffer and skipping re-reads of the same block. It does seem to be a good idea to make a special case of the VTOC sector and always re-read that one, just in case we change disks after writing the VTOC as the last operation on the old disk.

Line by Line

In the INSTALL routine we first make sure there is a Protocol Converter in the slot this RWTS expects. If so, we patch in the JMP to our code near the beginning of the normal RWTS and disable INIT by patching an RTS instruction at the beginning of the command handler. MOVE then puts our routine into place at \$BEAF and looks up the PC entry point into the ROMS and installs that address into the instructions that call the interface card. NO.PC provide an error message if we can't find a PC. The ID.TABLE has the bytes which mark a PC interface, interspersed with \$FFs so we can use the same index for the ROM and the table.

To boldly go at speeds no Apple has gone before.



Get TransWarp™. The fastest accelerator you can buy for your Apple™ IIe, II, or II+.

Computing at warp speed!

It's an experience you shouldn't miss. And with TransWarp, you won't have to. Because TransWarp will run your software up to 3.6 times faster — leaving other accelerators in the stardust!

No more yawning while your Apple™ slowly rearranges text or calculates spreadsheets. With 256K of ultra-fast RAM, TransWarp speeds up *all* Apple software — including AppleWorks, SuperCalc 3a, VisiCalc, and all educational software, graphics and games. And it's compatible with all standard peripheral cards (such as RamWorks II and Apple memory cards), Profile and Sider hard disks, 3½" UniDisks, 80-column cards, modems, clock cards, mice and more! You name it, TransWarp accelerates it. There's even a 16 bit upgrade chip available should 16 bit software become available for the Apple.



"I recommend Applied Engineering products wholeheartedly."

*Steve Wozniak, the creator
of Apple Computer*

An important difference.

TransWarp's not the only speedup card on the market. But it's the only one that accelerates your Apple's main memory, ROM *and* auxiliary memory. And with more and more programs residing in auxiliary memory, buying anyone else's accelerator makes less and less sense. TransWarp even works with most DMA devices including the Swift™ card.

There's one more difference. Since TransWarp doesn't use memory caching, it accelerates *all* software — and not just most of it.

A cinch to use.

Simply plug TransWarp into any slot in your Apple II, II+ or IIe — including slot 3 in the IIe. Instantly you'll be computing at speeds you only dreamed about before. And should you ever wish

to run at normal speed, simply press the ESC key while turning your Apple on.

Since TransWarp is completely transparent, you won't need pre-boot disks or special software. It's ready to go right out of the package!

Speed = Productivity

Imagine the productivity gains you'll achieve when your programs are running over three times faster. TransWarp is so powerful, your Apple will make IBM PCs™ and even ATs™ look like slowpokes — whether you're planning taxes, plotting charts or playing games! Take a look at a few of the features that set TransWarp apart:

- 3.6 MHz 65C02
- 256K of ultra-fast on-board RAM
- Accelerates main *and* auxiliary memory
- Low power consumption for cool operation
- Totally transparent operation with all software
- Plugs into any slot, including slot 3 on the Apple IIe
- Accelerated 16 bit option available

Satisfaction guaranteed!

Give *your* Apple the TransWarp advantage. With our risk-free 15-day money back guarantee, you have nothing to lose but wasted time. Call today!

TransWarp Accelerator
16 bit upgrade (may add later)

\$279
\$80

For fast response:

Call Applied Engineering, 9 a.m. to 11 p.m., 7 days at (214) 241-6000. MasterCard, VISA and C.O.D. welcome. Texas residents add 5.5% sales tax. Add \$10.00 if outside U.S.A.

Or mail check or money order to Applied Engineering, P.O. Box 798, Carrollton, TX 75006

AE Applied Engineering
The Apple enhancement experts.

P.O. Box 798, Carrollton, TX 75006 (214) 241-6000

The meat of the program begins at MY.RWTS. We enter here with slot*\$10 in the A register so we can check to see if we need to handle this call. If not we execute the instructions we overwrote with the JMP and go back to the normal RWTS. If is is our call, the first thing we do at MINE is check to see if we handled the last RWTS call as well. If so, all is well, but if normal RWTS was used last then it clobbered the buffer at \$BB00. We therefore trash LAST.BLOCK so the tests down at CHECK.FOR.RE.READ will be forced to read a new block.

SET.BLOCK transforms the requested track and sector into a block number, in the process setting carry to indicate whether we want the high or low half of the block. SET.POINTERS then creates two pointers for MY.BUFFER and IOB.BUFFER, using that carry bit along the way. At SET.DRIVE we check which drive is called for and modify BLOCK to read the other half of the diskette if it says drive 2. While we're at it, we plug the drive number into the volume number found, so it will appear as the volume number in a CATALOG. SET.COMMAND gets the command and makes sure it's either READ or WRITE. Anything else becomes a NOP.

At CHECK.FOR.RE.READ we compare the block number requested with the number of the block in the buffer and if they're different we go on to read the new block. If we already have the block we need, CHECK.FOR.VTOC double-checks to see if it's a VTOC we're reading. If so, we need to re-read it anyway, in case it's now a different disk in the drive. Once all that rigamarole is out of the way, the eight bytes at READ are all it takes to actually read the block!

At SKIP.READ we get the command again. (I just noticed that we can move the SET.COMMAND code to this point, since doing an extra READ won't hurt anything, even if the command is bad. That way we can eliminate MY.COMMAND and its STA and LDA instructions. Furthermore, changing the CMP #2 to an LSR and changing the BEQ to a BCC shaves out another byte, for a total of five fewer bytes. There's always more space to be found!) If the command is a READ then READ.MOVE.BUFFER copy MY.BUFFER into the IOB's buffer and we're done. If it's a WRITE, WRITE.MOVE.BUFFER copies the other way, from the IOB buffer into mine, and then calls the ROM to write out the block. Then GOOD.EXIT clears carry and loads a return code of zero before branching to the end. ERROR.EXIT loads up either WRITE PROTECT or DRIVE ERROR and sets carry before returning to the caller.

FORMAT 3.5 ---

Since we threw away INIT to fit all this inside of DOS, and since the standard INIT wouldn't put enough VTOC or CATALOG space on the disk, we're also going to need a special FORMAT program.

There are two stages in the process of formatting a disk: initializing all the tracks with address information; and writing the VTOC, empty catalog track, and boot program. Initializing a Protocol Converter device is easy, just call the

PC and let it do all the work. Then we can use our nice new RWTS to write all the rest of the necessary data. Just be sure that RWTS 3.5 is installed before calling FORMAT 3.5.

Since this catalog track is 31 sectors long there is room for 217 files instead of the normal 105. Other than the length, the structure is exactly the same as a normal DOS catalog. The differences in the VTOC are bytes \$34-35, the number of tracks per disk and sectors per track, and the bitmap. The bitmap skips tracks \$0 and \$11, fills all four bytes per track rather than alternate pairs, and extends all the way to the end of the sector.

The boot program here is just a quick message. I hope to have a real boot loader ready for next month's AAL.

```

1000 *SAVE S.UNIDISK RWTS
1010 *-----
05- 1020 UNIDISK.SLOT .EQ 5
1030
26- 1040 MY.COMMAND .EQ $26
3C- 1050 MY.BUFFER.POINTER .EQ $3C
3E- 1060 IOB.BUFFER.POINTER .EQ $3E
48- 1070 IOB.PTR .EQ $48
1080
BB00- 1090 MY.BUFFER .EQ $BB00
1100
BD12- 1110 PATCH.POINT .EQ $BD12
BD15- 1120 PATCH.RETURN .EQ $BD15
1130
C500- 1140 PC.DISPATCH .EQ UNIDISK.SLOT*100+$C000
1150
FDDA- 1160 PRBYTE .EQ $FDDA
FDED- 1170 COUT .EQ $FDED
1180 *-----
1190 .OR $803
1200 .IF RWTS 3.5
1210
1220 INSTALL
0803- A2 06 1230 LDX #6 make sure we have a
0805- BD 61 08 1240 .1 LDA ID.TABLE,X protocol converter
0808- DD 01 C5 1250 CMP UNIDISK.SLOT*100+$C001,X
080B- D0 31 1260 BNE NO.PC
080D- CA 1270 DEX
080E- CA 1280 DEX
080F- 10 F4 1290 BPL .1
1300
0811- A9 4C 1310 LDA #$4C patch in the JMP
0813- 8D 12 BD 1320 STA PATCH.POINT to our code
0816- A9 AF 1330 LDA #MY.RWTS
0818- 8D 13 BD 1340 STA PATCH.POINT+1
081B- A9 BE 1350 LDA /MY.RWTS
081D- 8D 14 BD 1360 STA PATCH.POINT+2
0820- A9 60 1370 LDA #$60
0822- 8D 4F A5 1380 STA $A54F disable INIT
1390
0825- A0 E8 1400 MOVE LDY #IMAGE.SIZE+1 install our code
0827- B9 67 08 1410 .1 LDA IMAGE-1,Y
082A- 99 AE BE 1420 STA MY.RWTS-1,Y
082D- 88 1430 DEY
082E- D0 F7 1440 BNE .1
1450
0830- 18 1460 CLC
0831- AD FF C5 1470 LDA UNIDISK.SLOT*100+$COFF
0834- 69 03 1480 ADC #3 find protocol
0836- 8D 48 BF 1490 STA READ.CALL converter entry
0839- 8D 6A BF 1500 STA WRITE.CALL
083C- D0 0D 1510 BNE DONE ...always
1520

```




NEW !!!][IN A MAC: \$69.00

This Apple II emulator runs DOS 3.3 and PRODOS programs (including 6502 machine language routines) on a 512K Macintosh. All Apple II features are supported such as HI-RES/LO-RES graphics, 40/80 column text screens, language card and joystick. Also included: clock, RAM disk, keyboard buffer, on-screen HELP, access to the desk accessories and support for 4 logical disk drives. Package includes 2 MAC diskettes (PROGRAM holds emulation, communications and utility software, DATA holds DOS 3.3 and PRODOS system masters, including Applesoft and Integer BASIC) and 1 Apple II diskette (transfer software moves disk images to the MAC).

NEW !!! SCREEN.GEN: \$35.00

Develop HI-RES screens for your Apple II on a Macintosh. Don't be limited by MousePaint or other screen editors. Use MACPAINT (or any other application) on the MAC to create your Apple II screen. Then use SCREEN.GEN to transfer directly from the MAC to the Apple II (with SuperSerial card or equivalent). Package includes Apple II diskette with transfer software plus fully commented SOURCE code.

NEW !!! MIDI-MAGIC for Apple //c: \$49.00

Compatible with any MIDI equipped music keyboard, synthesizer, organ or piano. Package includes a MIDI-out cable (plugs directly into modem port - no modifications required!) and 6-song demo diskette. Large selection of digitized QRS player-piano music available for 19.00 per diskette (write for catalog). MIDI-MAGIC compatible with Apple II family using Passport MIDI card (or our own input/output card w/drum sync for only \$99.00).

FONT DOWNLOADER & EDITOR: \$39.00

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Use with any Word Processor program capable of sending ESC and control codes to printer. Switch back and forth easily between standard and custom fonts. Special printer functions (like expanded, compressed etc.) supported. HIRES screen editor lets you create your own characters and special graphics symbols. For Apple II, II+, //e. Specify printer: Apple Dot Matrix, C.Itoh 8510A (Prowriter), Epson FX 80/100, or OkiData 92/93.

* The Font Downloader & Editor for the Apple Imagewriter Printer. For use with Apple II, II+, //e (with SuperSerial card) and the Apple //c (with builtin serial interface).

* FONT LIBRARY DISKETTE #1: \$19.00 contains lots of user-contributed fonts for all printers supported by the Font Downloader & Editor. Specify printer with order.

DISASM 2.2e : \$30.00 (\$50.00 with SOURCE Code)

Use this intelligent disassembler to investigate the inner workings of Apple II machine language programs. DISASM converts machine code into meaningful, symbolic source compatible with S-C, LISA, ToolKit and other assemblers. Handles data tables, displaced object code & even provides label substitution. Address-based triple cross reference generator included. DISASM is an invaluable machine language learning aid to both novice & expert alike. Don Lancaster says DISASM is "absolutely essential" in his ASSEMBLY COOKBOOK.

The 'PERFORMER' CARD: \$39.00 (\$59.00 with SOURCE Code)

Converts a 'dumb' parallel printer I/F card into a 'smart' one. Command menu eliminates need to remember complicated ESC codes. Features include perforation skip, auto page numbering with date & title. Includes large HIRES graphics & text screen dumps. Specify printer: MX-80 with Grafrax-80, MX-100, MX-80/100 with Grafraxplus, NEC 8092A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93.

'MIRROR' ROM: \$25.00 (\$45.00 with SOURCE Code)

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Basic modes: Dumb Terminal, Remote Console & Programmable Modem. Features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back, printer buffer, 80 col card & shift key mod support.

RAM/ROM DEVELOPMENT BOARD: \$30.00

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip (6116 type RAM for program development or 2716 EPROM to keep your favorite routines on-line). Maps into \$Cn00-CnFF and \$C800-CFFF.

C-PRINT For The APPLE //c: \$69.00

Connect standard parallel printers to an Apple //c. C-PRINT plugs into the standard Apple //c printer serial port and into any printer having a standard 36 pin centronics-type parallel connector. Just plug in and print!

Unless otherwise specified, all Apple II diskettes are standard (not copy protected!) 3.3 DOS.

Avoid a \$3.00 handling charge by enclosing full payment with order.

VISA/MC and COD phone orders OK.

RAK-WARE 41 Ralph Road W. Orange N J 07052 (201) 325-1885



```

083E- A2 00 1530 NO.PC LDX #0
0840- BD 4E 08 1540 .1 LDA MESSAGES,X print an error message
0843- F0 06 1550 BEQ DONE
0845- 20 ED FD 1560 JSR COUT
0848- E8 1570 INX
0849- D0 F5 1580 BNE .1
084B- 4C D0 03 1590 DONE JMP $3D0
1600 #-----
1610 MESSAGES
1620 .HS 8D

084E- 8D 1620
084F- CE EF A0
0852- D0 C3 A0
0855- E9 EE A0
0858- F3 EC EF
085B- F4 A0 1630 .AS -/No PC in slot /
085D- B5 1640 .DA $$B0+UNIDISK.SLOT
085E- 87 8D 00 1650 .HS 878D00
1660 #-----

0861- 20 FF 00
0864- FF 03 FF
0867- 00 1670 ID.TABLE .HS 20.FF.00.FF.03.FF.00
1680 #
1690 # Protocol Converter ID Bytes
1700 #-----
0868- 1710 IMAGE .EQ #
1720 .PH $BEAF
1730 MY.RWTS
BEAF- C9 50 1740 CMP #UNIDISK.SLOT*$10
BEB1- F0 06 1750 BEQ MINE my call!
BEB3- AA 1760 TAX not mine, so do
BEB4- A0 0F 1770 LDY $$F patched-over code
BEB6- 4C 15 BD 1780 JMP PATCH.RETURN and go back
1790 #-----
1800 MINE
BEB9- A0 0F 1810 LDY $$F
BEBB- D1 48 1820 CMP (IOB.PTR),Y check previous slot
BEBD- F0 07 1830 BEQ SET.BLOCK same, so go on
BEBF- 91 48 1840 STA (IOB.PTR),Y set previous slot
BEC1- A9 FF 1850 LDA $$FF
BEC3- 8D 8C BF 1860 STA LAST.BLOCK trash LAST.BLOCK
1870
1880 SET.BLOCK
BEC6- A9 00 1890 LDA #0
BEC8- 8D 8A BF 1900 STA BLOCK+1
BECB- A0 04 1910 LDY #4
BEDC- B1 48 1920 LDA (IOB.PTR),Y get track
BEDF- 0A 1930 .1 ASL
BED0- 2E 8A BF 1940 ROL BLOCK+1 #16
BED3- 88 1950 DEY
BED4- D0 F9 1960 BNE .1
BED6- 8D 89 BF 1970 STA BLOCK
BED9- A0 05 1980 LDY #5
BEDB- B1 48 1990 LDA (IOB.PTR),Y get sector
BEDD- 4A 2000 LSR 72, odd/even into carry
BEDE- 0D 89 BF 2010 ORA BLOCK
BEE1- 8D 89 BF 2020 STA BLOCK
2030
2040 SET.POINTERS
BEE4- A9 00 2050 LDA #MY.BUFFER
BEE6- 85 3C 2060 STA MY.BUFFER.POINTER
BEE8- A9 BB 2070 LDA /MY.BUFFER
BEEA- 69 00 2080 ADC #0 carry sets hi/lo half of buffer
BEEC- 85 3D 2090 STA MY.BUFFER.POINTER+1
BEEE- A0 08 2100 LDY #8
BEF0- B1 48 2110 LDA (IOB.PTR),Y get IOB buffer
BEF2- 85 3E 2120 STA IOB.BUFFER.POINTER
BEF4- C8 2130 INY
BEF5- B1 48 2140 LDA (IOB.PTR),Y
BEF7- 85 3F 2150 STA IOB.BUFFER.POINTER+1
2160
2170 SET.DRIVE
BEF9- A0 02 2180 LDY #2
BEFB- B1 48 2190 LDA (IOB.PTR),Y get drive
BEFD- A0 10 2200 LDY $$10
BEFF- 91 48 2210 STA (IOB.PTR),Y set previous drive
BF01- 88 2220 DEY
BF02- 88 2230 DEY
BF03- 91 48 2240 STA (IOB.PTR),Y set previous volume

```

With Z-80 Plus,TM run CP/M[®]—the largest body of software in existence.



*Now, get two computers in one,
and all the advantages of both.*

Enter the CP/M world with the new Z-80 Plus card from Applied Engineering, and introduce your Apple II[®] or II +[®] to the thousands of CP/M programs. Only the Z-80 Plus comes standard with the new 4.0 software, the most advanced system ever for running CP/M programs.

The new 4.0 boasts advanced features like built-in disk emulation for popular memory expansion boards, boosting both system speed and storage capacity. And menu-driven utilities that let you get to work faster. The Z-80 Plus also lets you run older CP/M programs — all the way down to Version 1.6 (2.2 is the most popular).

The Z-80 Plus is the only card on the market capable of accessing more than 64K in an Apple II[®]. If you have an extended 80-column card, all 128K is usable, and if you have RamWorks, up to 1088K is available.

Each Z-80 Plus comes with our CP/M Ram Drive software, enabling II[®] owners to use an extended 80-column card or a RamWorks card as a high-speed Ram disk which runs CP/M software up to *nearly times faster*. So packages like WordStar and dBASE II run at blinding speed.

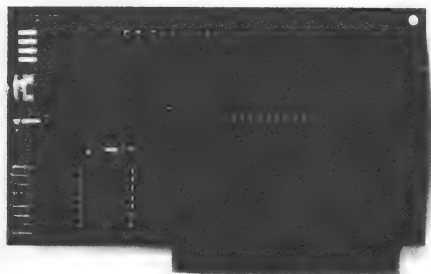
Simply plug the Z-80 Plus into any slot in your Apple. You'll get the benefits of two computers in one — all at an unbelievably low price (only \$139).

- Fully compatible with ALL CP/M software
- Fully compatible with most hard disks, including Corvus and the Sider
- Fully compatible with Microsoft disks (no pre-boot required)
- Specifically designed for high speed operation in the Apple II[®] (runs just as fast in the Apple II +[®] and Franklin)
- Runs WordStar, dBASE II, Turbo Pascal, Fortran-80, Peachtree and ALL other CP/M software with no pre-boot
- Semi-custom I.C. and low parts count allows Z-80 Plus to fly through CP/M programs with extremely low power consumption (we use the Z-80B)
- Does EVERYTHING other Z-80 boards do, plus Z-80 interrupts
- Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering, MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Timemaster H.O.TM, the only clock that displays time and date on AppleWorksTM screens and files.



*Now, get all the features of
all the competition combined!*

It's the smart way to put the time and date on your Apple II +[®] or II[®]. Because only the Timemaster H.O. packs *ALL* the features of all the competition *combined*, including leap year, year (not just in PRO-DOS), month, date, day of week, hours, minutes, seconds and milliseconds. It's totally PRO-DOS, DOS 3.3, PASCAL and CP/M compatible. And of course, it works better than any other clock with AppleWorks.

If you're using or writing software for other clock cards, you're still covered. Because the H.O. will *automatically* emulate them. And the Timemaster H.O. adds 14 new commands to BASIC. The H.O. even comes complete with two disks full of sample programs, including a computerized appointment book, a DOS dating program, interrupt programs, and over 30 programs that others charge extra for — or don't even offer.

As a low-cost option, you can add true BSR remote control to the H.O., giving you remote control of up to 16 lights and appliances in your home or office.

- Fully PRO-DOS and DOS 3.3, CP/M and PASCAL compatible
- Time in hours, minutes, seconds and milliseconds (the ONLY PRO-DOS compatible card with millisecond capability), date with year, month, day of week and leap year
- 24-Hour military format or 12-hour AM/PM format
- Eight software controlled interrupts so you can run two programs at the same time (many examples included)
- Allows AppleWorks to time and date stamp all data automatically
- The only clock card that displays time and date on the AppleWorks screen
- Five year warranty

Clock price	\$129.00
BSR option (may be added later)	\$ 49.00

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering, MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

BF05-	4A	2250	LSR		
BF06-	B0 10	2260	BCS SET.COMMAND	.CS. if D1	
BF08-	AD 89 BF	2270	LDA BLOCK	add 800 to BLOCK if D2	
BF0B-	69 20	2280	ADC #800		
BF0D-	8D 89 BF	2290	STA BLOCK		
BF10-	AD 8A BF	2300	LDA BLOCK+1		
BF13-	69 03	2310	ADC /800		
BF15-	8D 8A BF	2320	STA BLOCK+1		
		2330			
		2340	SET.COMMAND		
BF18-	A0 0C	2350	LDY #C		
BF1A-	B1 48	2360	LDA (IOB.PTR),Y	get command	
BF1C-	F0 53	2370	BEQ GOOD.EXIT		
BF1E-	C9 03	2380	CMP #3	exit if not READ or WRITE	
BF20-	B0 4F	2390	BCS GOOD.EXIT		
BF22-	85 26	2400	STA MY.COMMAND	save command	
		2410			
		2420	CHECK.FOR.RE.READ		
BF24-	A2 00	2430	LDX #0	zero the flag	
BF26-	A0 01	2440	LDY #1	check two bytes	
BF28-	B9 89 BF	2450	LDA BLOCK,Y		
BF2B-	D9 8C BF	2460	CMP LAST.BLOCK,Y	compare	
BF2E-	F0 04	2470	BEQ .2	same, so go on	
BF30-	E8	2480	INX	different, so flag it	
BF31-	99 8C BF	2490	STA LAST.BLOCK,Y	and store new value	
BF34-	88	2500	DEY		
BF35-	10 F1	2510	BPL .1	now do low bytes	
BF37-	8A	2520	TXA	check the flag	
BF38-	D0 0D	2530	BNE READ	if different, go read	
		2540			
		2550	CHECK.FOR.VTOC		
BF3A-	A0 05	2560	LDY #5		
BF3C-	B1 48	2570	LDA (IOB.PTR),Y	get sector	
BF3E-	D0 0F	2580	BNE SKIP.READ	non-zero isn't VTOC	
BF40-	88	2590	DEY		
BF41-	B1 48	2600	LDA (IOB.PTR),Y	get track	
BF43-	C9 11	2610	CMP #\$11		
BF45-	D0 08	2620	BNE SKIP.READ	not \$11 isn't VTOC	
		2630			
BF47-	20 00 C5	2640	READ JSR PC.DISPATCH		
BF48-		2650	READ.CALL .EQ #-2		
BF4A-	01	2660	.DA #1	READ	
BF4B-	85 BF	2670	.DA PARMLIST		
BF4D-	B0 27	2680	BCS ERROR.EXIT		
		2690			
		2700	SKIP.READ		
BF4F-	A5 26	2710	LDA MY.COMMAND	check command	
BF51-	C9 02	2720	CMP #2		
BF53-	F0 0B	2730	BEQ WRITE.MOVE.BUFFER		
		2740			
		2750	READ.MOVE.BUFFER		
BF55-	A0 00	2760	LDY #0		
BF57-	B1 3C	2770	LDA (MY.BUFFER.POINTER),Y		
BF59-	91 3E	2780	STA (IOB.BUFFER.POINTER),Y		
BF5B-	C8	2790	INY		
BF5C-	D0 F9	2800	BNE .1		
BF5E-	F0 11	2810	BEQ GOOD.EXIT	...always	
		2820			
		2830	WRITE.MOVE.BUFFER		
BF60-	A0 00	2840	LDY #0		
BF62-	B1 3E	2850	LDA (IOB.BUFFER.POINTER),Y		
BF64-	91 3C	2860	STA (MY.BUFFER.POINTER),Y		
BF66-	C8	2870	INY		
BF67-	D0 F9	2880	BNE .1		
		2890			
BF69-	20 00 C5	2900	WRITE JSR PC.DISPATCH		
BF6A-		2910	WRITE.CALL .EQ #-2		
BF6C-	02	2920	.DA #2	WRITE	
BF6D-	85 BF	2930	.DA PARMLIST		
BF6F-	B0 05	2940	BCS ERROR.EXIT		
		2950			
		2960	GOOD.EXIT		
BF71-	18	2970	CLC		
BF72-	A9 00	2980	LDA #0		
BF74-	F0 0A	2990	BEQ EXIT	...always	
		3000			

```

BF76- C9 2B 3010 ERROR.EXIT
BF78- F0 03 3020 CMP ##2B write protect?
BF7A- A9 40 3030 BEQ .1
3040 LDA ##40 make everything else DRIVE ERROR
BF7C- 2C 3050 .HS 2C
BF7D- A9 10 3060 .1 LDA ##10
BF7F- 38 3070 SEC
3080
BF80- A0 0D 3090 EXIT LDY ##D
BF82- 91 48 3100 STA (IOB.PTR),Y save return code
BF84- 60 3110 RTS
3120 #-----
3130 PARMLIST
BF85- 03 3140 .DA #3 3 parameters
BF86- 01 3150 .DA #1 unit number
BF87- 00 BB 3160 .DA MY.BUFFER buffer address
BF89- 3170 BLOCK .BS 3 block number
3180
BF8C- FF FF 3190 LAST.BLOCK .HS FFFF
3200 #-----
BF8E- 3210 .BS $BF97-*
3220 .EP
094F- 3230 IMAGE.END .EQ #-1
E7- 3240 IMAGE.SIZE .EQ IMAGE.END-IMAGE

```

```

1000 *SAVE S.FORMAT.UNIDISK
1010 #-----
05- 1020 UNIDISK.SLOT .EQ 5
1030
03D9- 1040 RWTS .EQ $3D9
1050
C500- 1060 PC.DISPATCH .EQ UNIDISK.SLOT*100+$C000
1070
FC58- 1080 HOME .EQ $FC58
FDED- 1090 COUT .EQ $FDED
1100 #-----
1110 .OR $803
1120 .TF FORMAT.UNIDISK
1130
0803- 18 1140 FORMAT CLC
0804- AD FF C5 1150 LDA UNIDISK.SLOT*100+$COFF
0807- 69 03 1160 ADC #3
0809- 8D 0D 08 1170 STA PC.CALL
080C- 20 00 C5 1180 JSR PC.DISPATCH format the disk
080D- 1190 PC.CALL .EQ #-2
080F- 03 1200 .DA #3
0810- 9E 08 1210 .DA PC.PARMS
0812- B0 7F 1220 BCS ERROR
0814- A9 02 1230 LDA #2
0816- 8D A2 08 1240 STA DRIVE do drive 2 first
1250
1260 DO.CATALOG
0819- 20 94 08 1270 JSR CLEAR.BUFFER
081C- A9 11 1280 LDA ##11
081E- 8D A4 08 1290 STA TRACK
0821- 8D 00 09 1300 STA MY.BUFFER+1 link pointer
0824- A0 1F 1310 LDY ##1F
0826- 8C A5 08 1320 .1 STY SECTOR
0829- 88 1330 DEY
082A- D0 03 1340 BNE .2
082C- 8C 00 09 1350 STY MY.BUFFER+1 mark end of catalog
082F- 8C 01 09 1360 .2 STY MY.BUFFER+2 link pointer
0832- 20 89 08 1370 JSR CALL.RWTS
0835- AC A5 08 1380 LDY SECTOR
0838- 88 1390 DEY
0839- D0 EB 1400 BNE .1 and go back for more
083B- 8C A5 08 1410 STY SECTOR
1420

```

```

1430 DO.VTOC
083E- 20 94 08 1440 JSR CLEAR.BUFFER
0841- A2 00 1450 LDX #0
0843- BC B1 08 1460 .1 LDY VTOC.INDEXES,X
0846- BD BC 08 1470 LDA VTOC.VALUES,X
0849- 99 FF 08 1480 STA MY.BUFFER,Y set VTOC header info
084C- E8 1490 INX
084D- E0 0B 1500 CPX #ENTRY.COUNT
084F- 90 F2 1510 BCC .1
0851- AD A2 08 1520 LDA DRIVE use drive # for volume
0854- 8D 05 09 1530 STA MY.BUFFER+6
0857- A9 FF 1540 LDA #$FF
0859- C8 1550 INY
085A- C8 1560 .2 INY skip a track in bitmap
085B- C8 1570 INY
085C- C8 1580 INY
085D- C8 1590 INY
085E- 99 FF 08 1600 .3 STA MY.BUFFER,Y mark free
0861- C8 1610 INY
0862- F0 06 1620 BEQ .4 leave if done
0864- C0 7C 1630 CPY #$7C track $11?
0866- F0 F2 1640 BEQ .2 yes, skip it
0868- D0 F4 1650 BNE .3 no, go on
086A- 20 89 08 1660 .4 JSR CALL.RWTS
086D- CE A2 08 1670 DEC DRIVE now go back and
0870- D0 A7 1680 BNE DO.CATALOG do drive one
1690
1700 DO.BOOT.SECTOR
0872- EE A2 08 1710 INC DRIVE that was drive one,
0875- 20 94 08 1720 JSR CLEAR.BUFFER so write a boot sector
0878- 8D A4 08 1730 STA TRACK A = 0
087B- 8D A5 08 1740 STA SECTOR
087E- A0 38 1750 LDY #BOOT.SIZE
0880- B9 C7 08 1760 .1 LDA BOOT.IMAGE,Y install the image
0883- 99 FF 08 1770 STA MY.BUFFER,Y
0886- 88 1780 DEY
0887- 10 F7 1790 BPL .1 fall into CALL.RWTS
1800 -----
1810 CALL.RWTS
0889- A9 08 1820 LDA /IOB
088B- A0 A0 1830 LDY #IOB
088D- 20 D9 03 1840 JSR RWTS
0890- B0 01 1850 BCS ERROR
0892- 60 1860 RTS
0893- 00 1870 ERROR BRK
1880 -----
1890 CLEAR.BUFFER
0894- A0 00 1900 LDY #0
0896- 98 1910 TYA
0897- 99 FF 08 1920 .1 STA MY.BUFFER,Y
089A- C8 1930 INY
089B- D0 FA 1940 BNE .1
089D- 60 1950 RTS
1960 -----
089E- 01 1970 PC.PARMS .DA #1 one parm
089F- 01 1980 .DA #1 unit one
1990 -----
2000 IOB .DA #1
08A1- 50 2010 SLOT .DA #UNIDISK.SLOT#$10
08A2- 2020 DRIVE .BS 1
08A3- 00 2030 VOL .DA #0
08A4- 2040 TRACK .BS 1
08A5- 2050 SECTOR .BS 1
08A6- FB B7 2060 DCT .DA $B7FB
08A8- FF 08 2070 BUFFER .DA MY.BUFFER
08AA- 2080 .BS 1
08AB- 00 2090 .DA #0
08AC- 02 2100 COMAND .DA #2 write
08AD- 2110 RETURN .BS 1
08AE- 2120 P.VOL .BS 1
08AF- 2130 P.SLOT .BS 1
08B0- 2140 P.DRIV .BS 1
2150 -----
08B1- 00 01 02
08B4- 03 27 30
08B7- 31 34 35
08BA- 36 37
0B- 2160 VTOC.INDEXES .HS 00.01.02.03.27.30.31.34.35.36.37
2170 ENTRY.COUNT .EQ *-VTOC.INDEXES

```

```

08BC- 04 11 1F
08BF- 03 7A 11
08C2- 01 32 20
08C5- 00 01
2180 VTOC.VALUES .HS 04.11.1F.03.7A.11.01.32.20.00.01
2190 *-----
2200 BOOT.IMAGE
2210 .PH $800
2220 BOOT .HS 01
2230 JSR HOME
2240 LDY #0
2250 .1 LDA MESSAGE,Y
2260 BEQ .2
2270 JSR COUT print message
2280 INY
2290 BNE .1
2300 .2 BEQ .2 and hang...
2310
2320 MESSAGE
2330 .HS 8D8D8D
0813- 8D 8D 8D
0816- D3 EF F2
0819- F2 F9 AC
081C- A0 E3 E1
081F- EE A7 F4
0822- A0 E2 EF
0825- EF F4 A0
0828- C4 CF D3
082B- A0 E8 E5
082E- F2 E5 A0
0831- F9 E5 F4
0834- AE
0835- 8D 87 00
2340 .AS -/Sorry, can't boot DOS here yet./
2350 .HS 8D8700
2360 .EP
38- 2370 BOOT.SIZE .EQ *-BOOT.IMAGE
2380 *-----
2390 MY.BUFFER

```

We Make Measurement And Control Easy!

12 BIT, 16 CHANNEL, PROGRAMMABLE GAIN A/D

- All new 1984 design incorporates the latest in state-of-art I.C. technologies.
- Complete 12 bit A/D converter, with an accuracy of 0.025%
- 16 single-ended channels (single ended means that your signals are measured against the Apple's GND) or 8 differential channels. Meet all the signals you will measure are single ended
- 9 software programmable full scale ranges, any of the 16 channels can have any range at any time. Under program control, you can select any of the following ranges: ± 10 volts, ± 5 V, ± 2.5 V, ± 1.0 V, ± 500 mV, ± 250 mV, ± 100 mV, ± 50 mV, or ± 25 mV.
- Very fast conversion (25 microseconds)
- Analog input resistance greater than 1,000,000 ohms
- Laser-trimmed scaling resistors
- Low power consumption through the use of CMOS devices
- The user connector has ± 12 and ± 12 volts on it so you can power your sensors
- Only elementary programming is required to use the A/D
- The entire system is on one standard size plug-in card that fits neatly inside the Apple
- System includes sample programs on disk

PRICE \$319

A few applications may include the monitoring of ● flow ● temperature ● humidity ● wind speed ● wind direction ● light intensity ● pressure ● RPM ● soil moisture and many more.

A/D & D/A

- A/D & D/A features:
 - Single PC card
 - 8 channels A/D
 - 8 channels D/A
 - Superfast conversion time
 - Very easy programming
 - Many analog ranges
 - Manual contains sample applications

A/D SPECIFICATIONS

- 0.1% accuracy
- On-board memory
- Fast conversion (0.78 MS per channel)
- A/D process is totally transparent to Apple (looks like memory)
- User programmable input ranges are 0 to 10 volts, 0 to 5, -5 to $+5$, -2.5 to $+2.5$, -5 to 0, -10 to 0.

The A/D process takes place on a continuous channel sequencing basis. Data is automatically transferred to its proper location in the on-board RAM. No A/D converter could be easier to use.

D/A SPECIFICATIONS

- 0.1% accuracy
- On-board memory
- On-board output buffer amps can drive 5 MA
- D/A process is totally transparent to the Apple (just poke the data)
- Fast conversion (0.01 MS per channel)
- User programmable output ranges are 0 to 5 volts and 0 to 10 volts

The D/A section contains 8 digital to analog converters, with output buffer amplifiers and all interface logic on a single card. On card latches are provided for each of the eight D/A converters. No D/A converter could be easier to use. The on-board amplifiers are laser-trimmed during manufacture, thereby eliminating any requirement for off-set nulling.

PRICE \$199

SIGNAL CONDITIONER

Our 8 channel signal conditioner is designed for use with both our A/D converters. This board incorporates 8 I.I.T. op-amps, which allow almost any gain or offset. For example, an input signal that varies from 2.00 to 2.15 volts or a signal that varies from 0 to 50 mV can easily be converted to 0-10V output for the A/D.

The signal conditioner's outputs are on a high quality 16 pin gold I.C. socket that matches the one on the A/D's so a simple ribbon cable connects the two. The signal conditioner can be powered by your Apple or from an external supply.

FEATURES

- 4 5" square for standard card cage and 4 mounting holes for standard mounting. The signal conditioner does not plug into the Apple, it can be located up to 1/2 mile away from the A/D.
- 22 pin 156 spacing edge card input connector (extra connectors are easily available i.e. Radio Shack).
- Large bread board area.
- Full detailed schematic included.

PRICE \$79

I/O 32

- Provides 4, 8-Bit programmable I/O Ports
- Any of the 4 ports can be programmed as an input or an output port
- All I/O lines are TTL (0-5 volt) compatible
- Your inputs can be anything from high speed logic to simple switches
- Programming is made very easy by powerful on-board firmware
- The I/O 32 is your best choice for any control application

The I/O manual includes many programs for inputs and outputs.

Some applications include:

Burglar alarm, direction sensing, use with relays to turn on lights, sound buzzers, start motors, control tape recorders and printers, use with digital joystick

PRICE \$89

Please see our other full page ad in this magazine for information on Applied Engineering's Interconnect Clock Card and other products for the Apple.

Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. PC boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products compatible with Apple II and IIe.

Applied Engineering's products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no-hassle three year warranty.

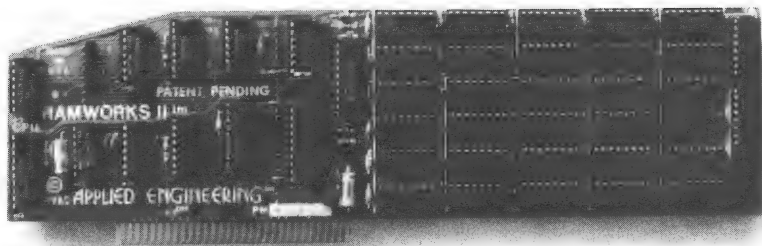
Texas Residents Add 5% Sales Tax
Add \$10.00 if Outside U.S.A.

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 798
Carrollton, TX 75006

Call (214) 241-6060
9 a.m. to 11 p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome
No extra charge for credit cards

RamWorks II®

The Best Selling, Most Compatible, Most Recommended, Most Expandable Card Available.



64K to 16 MEG! RamWorks II Is Number One.

It's simple, RamWorks II sells the most because it does the most.

The AppleWorks Amplifier.

While RamWorks II is recognized by all memory intensive programs, NO other expansion card comes close to offering the multitude of enhancements to AppleWorks that RamWorks II does. Naturally, you'd expect RamWorks II to expand the available desktop, after all Applied Engineering was a year ahead of everyone else *including Apple* in offering more than 55K in AppleWorks and we still provide the largest AppleWorks desktops available. But a larger desktop is just part of the story. Just look at all the AppleWorks enhancements that even Apple's own card does not provide and *only* RamWorks II does. With a 256K or larger RamWorks II, all of AppleWorks will automatically load itself into RAM dramatically increasing speed by eliminating all the time required to access the program disk drive. Now switch from word processing to spreadsheet to database at the speed of light with no wear on disk drives.

Only RamWorks II eliminates AppleWorks' internal memory limits, increasing the maximum number of records available from 1,350 to over 15,000. *Only* RamWorks II increases the number of lines permitted in the word processing mode from 2,250 to over 15,000. And *only* RamWorks II (256K or larger) offers a built-in printer buffer, so you won't have to wait for your printer to stop before returning to AppleWorks. Ram-

Works II even expands the clipboard. And auto segments large files so they can be saved on two or more disks.

RamWorks II, nothing comes close to enhancing AppleWorks so much.

The Most Friendly, Most Compatible Card Available.

Using RamWorks II couldn't be easier because it's compatible with more off-the-shelf software than any other RAM card. Popular programs like AppleWorks, Pinpoint, Catalyst, MouseDesk, HowardSoft, FlashCalc, The Spread Sheet, Managing Your Money, SuperCalc 3a, and MagiCalc to name a few (and *all* hardware add on's like ProFile and Sider hard disks). RamWorks II is even compatible with software written for Apple cards. But unlike other cards, RamWorks II plugs into the IIe auxiliary slot providing our super sharp 80 column text in a completely integrated system while leaving expansion slots 1 through 7 available for other peripheral cards.

Highest Memory Expansion.

Applied Engineering has always offered the largest memory for the IIe and RamWorks II continues that tradition by expanding to 1 full MEG on the main card using standard RAMs, more than most will ever need (1 meg is about 500 pages of text)...but if you do ever need more, RamWorks II has the widest selection of expander cards available. Additional 512K, 2 MEG, or multiple 16 MEG cards just snap directly onto RamWorks II by plugging into the

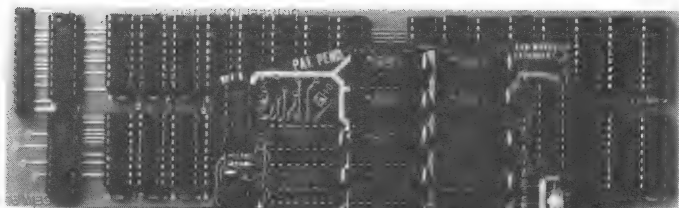
industry's only low profile (no slot 1 interference) fully decoded memory expansion connector. You can also choose non-volatile, power independent expanders allowing permanent storage for over 20 years.

It Even Corrects Mistakes.

If you've got some other RAM card that's not being recognized by your programs, and you want RamWorks II, you're in luck. Because all you have to do is plug the memory chips from your current card into the expansion sockets on RamWorks II to recapture most of your investment!

The Ultimate in RGB Color.

RGB color is an option on RamWorks II and with good reason. Some others combine RGB output with their memory cards, but that's unfair for those who don't need RGB *and* for those that do. Because if you don't need RGB Applied Engineering doesn't make you buy it, but if you want RGB output you're in for a nice surprise because the RamWorks II RGB option offers better color graphics plus a more readable 80 column text (that blows away any composite color monitor). For only \$129 it can be added to RamWorks II, giving you a razor sharp, vivid brilliance that most claim is the best they have ever seen. You'll also appreciate the multiple text colors (others only have green) that come standard. But the RamWorks II RGB option is more than just the ultimate in color output because unlike others, it's fully



16 MEG Expander

512K Expander

compatible with all the Apple standards for RGB output control, making it more compatible with off-the-shelf software. With its FCC certified design, you can use almost any RGB monitor because only the new RamWorks II RGB option provides both Apple standard and IBM standard RGB outputs (cables included). The RGB option plugs into the back of RamWorks II with no slot 1 interference (works on the original RamWorks, too) and remember you can order the RGB option with your RamWorks II or add it on at a later date.

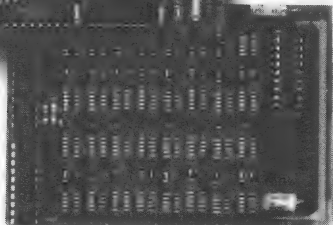
True 65C816 16 Bit Power.

RamWorks II has a built-in 65C816 CPU port for direct connection to our optional 65C816 card. The only one capable of linearly addressing more than 1 meg of memory for power applications like running the Lotus 1-2-3™ compatible program, VIP Professional. Our 65C816 card does not use another slot but replaces the 65C02 yet maintains full 8 bit compatibility.

Endorsed by the Experts.

Steve Wozniak, creator of the Apple Computer said "I wanted a memory card for my Apple that was fast, easy to use, and very compatible; so I bought RamWorks." A+ magazine said "Applied Engineering's RamWorks is a boon to those who must use large files with AppleWorks...I like the product so much that I am buying one for my own system." inCider magazine said "RamWorks II is the most powerful auxiliary slot memory card available for your IIe, and I rate it four stars...For my money, Applied Engineering's RamWorks II is king of the hill."

Apple experts everywhere are impressed by RamWorks II's expandability, versatility, ease of use, and the sheer power and speed that it adds to any IIe. With a RamWorks II in your Apple, you'll make IBM PCs and ATs look like slowpokes.

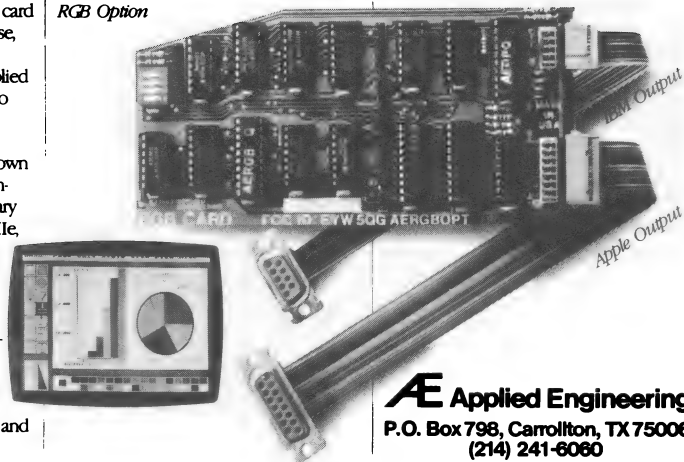


2 Meg Expander

It's Got It All

- 15 day money back guarantee
- 5 year hassle free warranty insures coverage no matter where you purchase
- Built-in super sharp 80 column display, (with or without RGB)
- Expandable to 1 MEG on main card
- Expandable to 16 meg with expander card, with NO slot 1 interference
- Can use 64K or 256K RAMs
- Powerful linear addressing 16 bit coprocessor port
- Automatic AppleWorks expansion up to 3017K desktop
- Accelerates AppleWorks
- Built-in AppleWorks printer buffer
- The only large RAM card that's 100% compatible with all IIe software

RGB Option



- RamDrive™ the ultimate disk emulation software included free
- Memory is easily partitioned allowing many programs to be in memory at once
- Compatible, RGB option featuring ultra high resolution color graphics and multiple text colors, with cables for both Apple and IBM type monitors
- Built-in self diagnostics software
- Lowest power consumption (patent pending)
- Takes only one slot (auxiliary) even when fully expanded
- Software industry standard
- Advanced Computer Aided Design
- Used by Apple Computer, Steve Wozniak and virtually all software companies
- Displays date and time on the AppleWorks screen with any PRO-DOS compatible clock
- Much, much more!

RamWorks II with 64K	\$179
RamWorks II with 256K	\$219
RamWorks II with 512K	\$269
RamWorks II with 1 MEG	\$369
RamWorks II with 1.5 MEG	\$539
RamWorks II with 3 to 16 MEG	CALL
65C816 16 Bit Card	\$159
RGB Option	\$129
256K Upgrade	\$ 50

RamWorks II. The industry standard for memory expansion of the Apple IIe. ORDER YOUR RamWorks II TODAY. 9 a.m. to 11 p.m. 7 days, or send check or money order to Applied Engineering. MasterCard, Visa and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 if outside U.S.A.

AE Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6080

As a long-time user of the S-C Macro Assembler, I have learned a few tricks to save a lot of aggravation. Sometimes I mistakenly erase the source program I have in memory with the "NEW" or "LOAD" command. The program is not actually gone; instead, the pointer to the start of the program is changed.

At one time, I would adjust the source pointer by hand until my program was restored, but this was slow and painful. So like all good hackers I now have a little program to find the start of a program and adjust the pointer automatically.

My "Find.Start" program searches through memory for a source line numbered 1000 and resets the source pointer to that line. The search begins at HIMEM and proceeds down until it finds line 1000 or address \$800.

The program itself is a simple search for the two-byte hex equivalent of 1000. On entry, the program starts the search at HIMEM and sets the "DONE.ONCE" flag so subsequent re-entries pick up the search where it last left off.

After the program stops, you can run it again to find the next lower source line numbered 1000. If several programs have been loaded into memory, you can run "Find.Start" several times to point to the start of each one.

The only way to start the search from HIMEM again is to re-load the program. It's not elegant, but does it really need to be?

In many instances, the next step is to re-construct the scrambled part of a program. This usually seems impossible, because the program's internal pointers will probably be scrambled and cause weird problems when editing.

Instead of fighting with the program (or hand-patching as I used to do), just use the handy "TEXT" command built into the assembler to create a text version of your program. Then enter the "AUTO" mode and "EXEC" the text version of your program back into memory. This will rectify all the internal pointers and leave you free to edit your program back into shape.

Perhaps that last paragraph is obvious, but I didn't think of it until recently. And we've had the "TEXT" command available for a long time!

```

1000 *SAVE FIND.START
1010 *-----
1020 *   SEARCH FROM HIMEM TO PP FOR LINE "1000"
1030 *   SET $CA,CB TO BEGINNING OF THAT LINE
1040 *-----
00- 1050 SRCP   .EQ $00,01
4C- 1060 HIMEM .EQ $4C,4D
CA- 1070 PP    .EQ $CA,CB
1080 *-----
1090      .OR $300
1100 *-----
1110 DO
0300- A6 CA 1120      LDX PP      IF NOT FIRST TIME,
0302- A5 CB 1130      LDA PP+1    START WHERE WE LEFT OFF
0304- 2C 45 03 1140      BIT DONE.ONCE.FLAG
0307- 30 08 1150      BMI .1      ...NOT FIRST TIME
```

```

1160 *---HAS TO BE A FIRST TIME-----
0309- 38 1170 SEC SET FLAG
030A- 6E 45 03 1180 ROR DONE.ONCE.FLAG
030D- A6 4C 1190 LDX HIMEM START AT TOP OF SOURCE AREA
030F- A5 4D 1200 LDA HIMEM+1
1210 *---STORE STARTING POINTER-----
0311- 86 00 1220 .1 STX SRCP
0313- 85 01 1230 STA SRCP+1
0315- 20 3C 03 1240 JSR DEC.SRCP
1250 *---SEARCH FOR "1000"-----
0318- 20 3C 03 1260 .2 JSR DEC.SRCP
031B- A5 01 1270 LDA SRCP+1
031D- C9 08 1280 CMP /$0800 DON'T SEARCH BEYOND $800
031F- 90 1A 1290 BCC .3 ...END OF SEARCH
0321- A0 00 1300 LDY #0
0323- B1 00 1310 LDA (SRCP),Y
0325- C9 E8 1320 CMP #1000 COMPARE LO-BYTE
0327- D0 EF 1330 BNE .2 ...NO, KEEP SCANNING
0329- C8 1340 INY ...MATCH, CHECK HI-BYTE
032A- B1 00 1350 LDA (SRCP),Y
032C- C9 03 1360 CMP /1000
032E- D0 E8 1370 BNE .2 ...NO, KEEP SCANNING
1380 *---FOUND IT, POINT PP TO IT-----
0330- 20 3C 03 1390 JSR DEC.SRCP BACK UP OVER BYTE COUNT
0333- A5 00 1400 LDA SRCP
0335- 85 CA 1410 STA PP
0337- A5 01 1420 LDA SRCP+1
0339- 85 CB 1430 STA PP+1
033B- 60 1440 .3 RTS
1450 *-----
1460 DEC.SRCP
1470 LDA SRCP
1480 BNE .1
1490 DEC SRCP+1
1500 .1 DEC SRCP
1510 RTS
1520 *-----
0345- 00 1530 DONE.ONCE.FLAG .HS 00
1540 *-----

```

RAMWORKS™

**ACCEPT NO SUBSTITUTES.
BECAUSE THERE AREN'T ANY.**

There's only one card like RamWorks. We've got the best hardware design. We supply the best software and we've got the best support from software companies.

If someone tempts you with an imitation, please get both sides of the story. You'll discover why RamWorks offers the best enhancements to AppleWorks and other programs, and at the lowest price.

GUARANTEED!

**214-241-6060
9 AM - 11 PM**



**"We Set the Standard"
APPLIED ENGINEERING**

More and Better Division by Seven.....Bob Sander-Cederlof

I can think of at least three good reasons we need a good subroutine for dividing by seven. We need it in computations involving the day of week. We need it in hi-res graphics programs to calculate the byte and bit for a particular pixel between 0 and 279 for normal hi-res, or between 0 and 559 for double hi-res. Lastly, the new protocol converter interface used in connection with the Unidisk 3.5 works with packets of up to 767 bytes which are made up of a number of 7-byte groups.

In looking through the assembly listing of the new //c ROMs, which come with the Unidisk 3.5 update, I noticed a divide-by-seven subroutine at \$CB45-CBAF. The code divides the buffer size, which can be up to \$2FF, by seven, and saves both the quotient and the remainder. The code looks too large and too slow and too complicated ... in other words, it looks like a challenging assignment. My transposition of the //c code follows, and as I count cycles it takes from 133 to 268 cycles depending on the value of the dividend. The code and tables take 71 bytes in the //c ROM.

While I was musing on the possibilities, Michael Hackney called me from Troy, New York. He wondered if we were interested in publishing his fast 65802 routine for dividing by seven. Michael uses his in a speedy double hi-res program. He divides values up to 559 (\$22F) by seven, keeping both the quotient and remainder, in 66 cycles. Michael's subroutine itself is short (37 bytes), but he uses a 140-byte table to achieve the speed. Adding another 84 bytes to the tables extends the range to handle dividends up to 895 (\$37F).

(In all the times and lengths given here, I am not counting the JSR-RTS cycles nor the RTS byte. I assume the code is critical enough that it would be placed in-line in actual use, rather than made into a JSR-called subroutine. I am also not counting any overhead I added to switch from 65802 mode to 6502 and back, as this was only added due to my test program being in 65802 mode. All of the subroutines use page zero for variable and temporary storage. They would be longer and slightly slower if the variables and temporaries were not in page zero.)

Yesterday I spent the whole day dividing by seven. I came up with two new subroutines: one for the 65802, and one for a normal 6502. They are both small and fast. First I tackled the 65802 version, and based in on multiplying by 1/7 as a binary fraction. This one came out 39 bytes long, executing in 64 cycles. This one used a fudge factor; the largest dividend it can handle is 594 (\$252). By using alternate code to extend the precision, numbers up to 895 (\$37F) can be handled. This one takes the same number of bytes, but 9 cycles longer.

Finally, I wrote a normal 6502 version. Strangely enough, it came out only 60 bytes long and only 76 cycles! Makes me wonder if I couldn't do better in the 65802, given another day or two. The 6502 version handles dividends up to 1023 (\$3FF). It would be two bytes shorter if the range was restricted to \$2FF.

Here is a table summarizing the size, timing, and dividend range for the various subroutines:

	bytes	cycles	dividend
//c ROM	71	133-268	0-\$2FF
Hackney 65802	177	66	0-\$22F
RBSC 65802-1	39	64	0-\$252
RBSC 65802-2	39	73	0-\$37F
RBSC 6502	60	76	0-\$3FF

The listing which follows includes all five versions, plus a testing program. The testing program runs through the entire range from \$3FF down to 0. After doing the division by the selected method, a check subroutine tests for a valid remainder (a number less than 7); it further tests that the quotient*7 + remainder = the original dividend. If not, the dividend, quotient, and remainder are all printed in hexadecimal. If they are correct, the next dividend is tried. A keyboard pausing subroutine allows you to stop the display momentarily and/or abort the test run.

Lines 1020-1060 control some conditional assembly which select which division method to use. By change the value of VERSION in line 1020 I can assemble any one of the four routines. I used the "CON" listing option in line 1180 (which is not itself listed: it is "1180 .LIST CON") so that you can see what the un-assembled lines of code are. Other conditional code at lines 1720-1860 and 4010-4050 selects options mentioned above.

Lines 1200-1540 control each test run. I wrote this program using 65802 instructions, although it would not be difficult to re-write it for a plain 6502. Lines 1210-1220 enter the 65802 Native Mode. and lines 1520-1530 leave it. It is VERY IMPORTANT to be sure you do not exit a program and return to normal Apple software while still in the Native Mode. The most fantastic things can happen if you forget!

Lines 1580-1950 are my 65802 version. This entire subroutine is executed in the 65802 native mode, with the M-bit set so the A-register operations are 16-bits. The value 1/7 in binary is .001001001001001...forever. Multiplying by than number should give the same answer as dividing by seven. It also has the surprising side effect that the three bits after the "quotient" portion of the product will be equal to the "remainder". The values of the fractions from 0/7 to 6/7 are just nice that way:

fraction	repeating decimal	same value in hex	the first three bits
0/7	.000000	.000	000
1/7	.142857..	.249..	001
2/7	.285714..	.492..	010
3/7	.428571..	.6DB..	011
4/7	.571428..	.924..	100
5/7	.714285..	.B6D..	101
6/7	.857142..	.DB6..	110

Wow! Isn't that neat? More justification for the numerologists who claim that seven is the "perfect" number.

Now it remains to find the most efficient way to multiply by that fraction. The method I came up with first forms the product for .01000001 (lines 1600-1670). Then I divide that result by 8, which is the product for .00001000001 (lines 1680-1700). Adding the two products in line 1710 gives me the product for .01001001001 (approximately $2/7$). Dividing that by two gives me an approximation for the division by seven. The code that follows in lines 1720-1800 is not assembled, because of the ".DO 0" line. What it does is extend the multiplication to include one more partial product. The shortest way I could think of to get that little number is demonstrated in the code you see. The extra precision makes my subroutine work for dividends up to \$37F. It fails above that value because of overflow during the multiplication. If I leave out the extra precision, the subroutine gets the wrong answers for some numbers at each end of the range. By adding a "fudge factor" (a trick learned in college laboratory assignments to force experimental results to fit the laws of science), I can make all the dividends up to \$252 work. The fudge factor adds \$000A for values in the A-register of \$8800 or more, and only \$0008 for values below \$8800.

Line 1870 is the division by two mentioned above. Lines 1880-1940 shift the first three bits of the remainder over to the correct position in the lower byte of the A-register. As I was writing the previous sentence, it suddenly struck me that the second set of three bits might be the same as the first set, if my multiplications happened to be precise enough. I went back to the assembler, changed line 1720 to ".DO 1" so the more precise version would assemble, and then replaced lines 1910-1930 with "1910 AND #7". Guess what! It worked! One byte shorter and four cycles faster! That makes it 38 bytes long, and only 69 cycles.

Next is my 6502 version, lines 1970-2370. The first four lines simply save the current state of the M and X bits, and the mode, and switch to 6502 emulation mode. They are matched by lines 2340-2360, which restore the mode and state. These will work regardless of what mode and state the machine was in when the subroutine was called. Since the subroutine would normally only be used in a 6502, you would leave out lines 1980-2010 and 2340-2360. I did not count them when timing the code. Back in December of 1984 I wrote in these pages of a nifty way to divide a one-byte value by seven. I used that method here, for dividing the low-order byte of the dividend. I then computed the remainder by multiplying the quotient by 7 and subtracting it from the dividend. Saving that quotient and remainder, I used a table lookup to determine the quotient and remainder of the high-order byte of the number. Since it could only have the values 0-3, the tables are very short. Then I add the two remainders together, modulo 7; and the two quotients, remembering the carry from the remainder if any.

Lines 2030-2170 are essentially the same as published in that December issue of AAL, except for the addition of lines 2130,

2140, and 2160. With those two lines I am saving a few steps in the multiplication by seven that I must do. Lines 2190-2200 finish the multiplication by seven, by adding the *2 and *4 values saved above. Lines 2210-2200 form the complement of the value. so I can subtract by adding. Normally a complement is formed by:

```
EOR #$FF
CLC
ADC #1
```

I do the same with two less bytes and cycles here by preceding the addition at line 2230 with SEC rather than the usual CLC. I saved a byte and two cycles by storing one less than the actual remainder in the table of remainders at line 2400.

Lines 2420-2640 are called to print out the results when they don't meet expectations. Notice lines 2430-2460 and 2610-2630, which make sure I am in the correct state and mode. The monitor routines will not work correctly in 16-bit state, and may not work correctly in 65802 Native mode.

Lines 2660-2920 check the results. The subroutine returns with carry clear if the quotient and remainder are correct, or carry set if they are not. I check both by multiplying the quotient by seven and adding the remainder to see if the result equals the dividend, and I also make sure the remainder is less than seven. It is possible to get an answer with the quotient one less than it should be and a remainder of 7, so I had to test the remainder.

The PAUSE routine checks to see if any key has been typed. If so, and if it is not a <RETURN>, it waits until another key is typed. Note that I had to set 8-bit mode, to prevent the softswitch at \$C011 from being switched. This also makes the CMP work properly. Otherwise the LDA \$C000 would get two copies of the same character in the two halves of the A-register.

Lines 3060-3540 are essentially the code from the new //c ROMs. I re-arranged it a little, to make a stand-alone routine within my test-bed, and I changed labels and variable names. Apple uses two sets of tables. One gives quotients and remainders for 0, \$100, and \$200 (the high byte of the dividend). The other gives quotients and remainders for 0, \$08, \$10, \$20, \$40, and \$80. A loop runs 5 times to add in the quotients and remainders for bits 3-7 of the dividend, and then fakes one more trip to add in the value of bits 0-2. Not efficient!

Michael Hackney's code is in lines 3560-4080. I'll quote from his letter.

"Apple hi-res graphics characteristically involve various calculations to determine the exact display address from a given X,Y pair. Typically, the vertical position (Y) base address is found by table look-up. The horizontal, or X, position is determined by dividing by 7 (since there are seven pixel bits per byte in the hi-res screen). The integer portion

1200 BAUD MODEMS

Coit Valley Computers has two modems for your every need. Both are top quality state-of-the-art 1200/300/110 baud Hayes™ compatible modems; which means your computer can send & receive data at lightning fast speeds! And automatically switch between 1200 and 300 baud to communicate with slower Apples. Since neither comes with software, we carry Ascii Express ProDOS at a low price of \$89.

AVATEX™ 1200 EXTERNAL STAND-ALONE MODEM

\$199.

- 100% plug in Modem for **Apple IIc or Macintosh** with proper cable (see below)
- Universal modem that only requires modem compatible serial card (or port), & cable, to plug into **Apple IIe, Apple II+, or IBM**
- Auto Answer, Auto Dial, Auto Redial, Auto Disconnect
- Full Bell 212A compatibility
- Automatically switches between 300 baud & 1200 baud incoming speeds
- Complete diagnostics & full complement of LEDS (TR, SD, RD, HS, MC, TM, RI)
- DATA/VOICE Button switches from talk to data transmission & back again
- FREE Compuserve offer & free access time. One year warranty.

CERMETEK APPLE-MATE™ 1200 INTERNAL MODEM

\$209.

- Internal 1200 baud modem for **Apple IIe or Apple II+**
- Only one card & takes only one slot w/ no external interface or power supply

- Built-in Super Serial Card equivalent
- 1200/300/110 baud operation and Bell 212A compatibility
- Built-in Speaker & Diagnostics
- Auto Dial, Auto Answer, & Auto Select. Two year warranty.

2400 BAUD MODEMS — Call

CABLES REQUIRED WITH AVATEX MODEMS

Apple IIc - Avatex Cable	\$ 22.
Apple IIe, II+ - Avatex Cable	25.
Macintosh - Avatex Cable	27.
IBM - Avatex Cable	23.

OTHER APPLE PERIPHERALS

IIe/II+ Serial Modem Card	\$ 99.
RGB Monitor for Apple IIe	324.
RGB Monitor to Apple Cable	24.
MultiRam RGB cards (facing page)	➡ ➡

With prices this low, how can you afford to be without a 1200 baud modem? Just the savings in connect time, will pay for the difference between a 300 & 1200 baud modem. You can get everything you need from Coit Valley Computers. Shipping on modems \$5-Ground/\$8-Air; monitors \$10. See terms on facing page.

Hayes, Avatex, Apple-Mate respective registered trademarks of Hayes Microcomputer Products, E + E DataComm, Cermetek Micro.

COIT VALLEY COMPUTERS • 14055 Waterfall Way, Dallas, TX 75240 • (214) 234-5047

6 Meg IIC/640k IIC

6 Meg 16-Bit IIC

Don't buy yesterday's card that doesn't offer battery backed-up RAM or 65C816 new Apple technology just because it's advertised a lot! You can buy Checkmate Technology's **State-Of-The-Art MULTIRAM RGB RAM CARD™** with **BATTERY BACKED-UP STATIC RAM** options that can load & save programs (like AppleWorks) for 10 years! It is a **FASTER & LESS EXPENSIVE REPLACEMENT FOR HARD DISKS**, is **USER EXPANDABLE TO 6 MEGABYTES**, compatible with all (100%) 3rd party software/hardware, has an optional real 16-Bit 65C816 slot saver Co-Processor card, sharp 80 columns, super Double Hi-Res, & **BUILT IN RGB™!** It's a direct substitute for Ramworks II™ or Apple Ext 80 column cards & has an amazing 5 year warranty! Unlike Ramworks II, MultiRam fits ALL (even Euro) Apples, can't interfere with slot 1 cards & has no soldered chips!

MultiRam RGB expands to 1 Meg main RAM + 3 Meg's piggyback RAM + 2 Meg's BATTERY BACKED-UP RAM. **MultiRam IIC** expands to 768k & can piggyback w/ MultiRam RGB. **A POSSIBLE 6 MEGABYTES IN ONE SLOT - MORE THAN RAMWORKS II & Filpster™.**

FREE APPLEWORKS EXPANDER SOFTWARE that loads ALL (even printer routines) or PARTS of AppleWorks, runs 30 x faster, increased Disksp over 3024k, auto-segments large files onto multiple disks, stores over 16,000 records! **FREE APPLEWORKS TIME/DAY/DATE ON-SCREEN, AUTO-COPY TO RAM, ULTRA-FAST PRODOS/DOS 3.3 RAM DISK & RAM TEST**, optional CP/M & Pascal Ram disk!

	MultiRam	MultiRam
	RGB	IIC
	Card	Card
64k MULTIRAM	169.	129.
128k MULTIRAM	179.	139.
320k MULTIRAM (256k + 64k FREE)	209.	175.
576k MULTIRAM (256k + 64k FREE)	248.	214.
832k MULTIRAM	275.	239.
1024k MULTIRAM	284.	249.
1280k MULTIRAM	490.	-
1536k MULTIRAM	525.	-
1792k MULTIRAM	559.	-

256k Memory Chips-1 yr warranty (B)	55.
Apple IIC Enhancement Kit	62.
Accelerator IIC-350% speedup card	222.
Accelerator IIC + Pinpoint (special)	259.
Clockworks Card (Thunder/Time HO™ comp)	89.
Pico™ Slimline Drive IIC, IIC, II+	178.
FD-100 Slimline Drive IIC, II+	115.
Pinpoint Program or Spell Checker (ea)	49.
65C816 EX Co-Processor Card	157.
RGB Connectors & Cables*	call

Terms: Add \$4-Ground or \$6-Air shipping & phone # to each U.S. card order (foreign orders/FPO/APO extra). Add 3% for MasterCard/Visa (include #/expir) & P.O.'s (3% 7 Net. 30). For fast delivery send Cashier's/Certified check, Money Order. C.O.D. (add \$5) & personal checks accepted (allow 16 days). Tex res add 6 1/2% tax.

MultiRam, Ramworks/Ramworks II/TimeMaster II H.O./Z-Ram, Pico, Filpster, respective trademarks of Checkmate Technology, Applied Engineering, WGE, Citech.

640K 16-Bit IIC

Checkmate Technology's **State-Of-The-Art IIC** cards easily expand your IIC up to 640k, are 100% compatible with all IIC software/hardware, & come with the **SAME FREE SOFTWARE as MULTIRAM IIC (see above)**. **MULTIRAM C** is non-upgradable, **MULTIRAM CX** can be upgraded with a real 65C816 kit to likely run software for the new Apple computer!

- **UNLIKE Z-RAM™, THERE ARE NO JUMPER WIRES, CLIPS TO ATTACH, SOLDERED CHIPS, OR DRIVE REMOVAL REQUIRED FOR INSTALLATION.**
- **USES ABOUT 50% LESS POWER** than Z-RAM causing less power supply strain or battery pack drain!
- **15 DAY MONEY BACK SATISFACTION GUARANTEE, 5 YR WARRANTY, & LOWER PRICES** - We sell IIC cards for much less & our software updates are FREE & AUTOMATIC, while others charge \$10 or more!

OUR LOWEST PRICE

256k MULTIRAM C (While supply lasts)	159.
512k MULTIRAM C (While supply lasts)	189.
256k MULTIRAM CX	185.
512k MULTIRAM CX	239.
65C816 CX Kit (\$10 less w/ card)	129.
VIP Professional w/ any 65C816	117.
IIC System Clock (Same as Applied Eng)	66.
CP/M & Pascal Ram Disk (ea)	20.

WHY BUY FROM COIT VALLEY COMPUTERS RATHER THAN SOME MAIL ORDER HOUSES? Only we offer an exclusive 15 day money back satisfaction guarantee, double software, more support, free automatic software updates, free 64k with each 256k/512k card. We know the products well, & we have them in stock. **CALL FOR DETAILS, CURRENT PRICES, QUANTITY DISCOUNTS, OR NEW FEATURES! SCHOOLS & GROUPS WELCOME.**

O R D E R F O R M		
COIT VALLEY COMPUTERS (214) 234-5047		
14055 Waterfall Way Dallas, Texas 75240		
NAME _____		
ADDRESS _____		
CITY _____ STATE _____ ZIP _____		
PHONE () _____		
SIGNATURE _____		16
QTY	DESCRIPTION	PRICE
MC/VISA	SHIPPING	
EXP	TOTAL	

COIT VALLEY COMPUTERS

14055 Waterfall Way

(214) 234-5047

Dallas, Texas 75240

of the division is the byte offset from the base address, and the remainder is the position in the byte. Brute calculation (which is slow for graphics routines) or table lookup (which takes a lot of space) is used to do the division. Table lookup is usually used in good graphics programs. Hi-res graphics require two 280-byte tables, one for quotient and one for remainder. Double hi-res requires tables twice as big. My interest in 65802/816 double-hi-res graphics drivers has prompted me to find a serviceable divide-by-seven which is quick and doesn't require more than one page of memory.

"The 65802/816 16-bit operations are ideally suited for this task. Larger numbers can be easily manipulated and table lookup can retrieve 2 bytes of data at once. My routine uses both of these techniques to perform its duty. It divides the original number by eight before doing any table lookup (this keeps the table smaller). Then it multiplies both the quotient and remainder retrieved from the table by 8. The resulting remainder is added to the original lower three bits (the ones shifted out when I divided by 8), and I look into the table again. The first quotient is added to the second quotient, and it is finished. The table only takes 140 bytes, storing quotients and remainders for numbers up to 69. Everything fits in a page with room to spare.

"As an extra bonus, I included a small routine which generates the table in situ. The area occupied by the table generator can be used for data storage once the table is built. It takes longer to load a table from disk than it does to compute one, and the generator disappears after use, so this is the best way to do it."

In order to get the greatest speed, Michael's table should all reside entirely in the same page of memory. That is why I included line 4100, which justifies the table to the beginning of the next page.

So here you have four great answers to the challenge. Now it's your turn!

```

1000 *SAVE BETTER DIV 7+
1010 *-----
01- 1020 VERSION .EQ 1
01- 1030 RBSC65802 .EQ 1
02- 1040 HACKNEY .EQ 2
03- 1050 TWO.C .EQ 3
04- 1060 RBSC6502 .EQ 4
1070 *-----
00- 1080 DIVIDEND .EQ 0,1
02- 1090 QUO.REM .EQ 2,3
04- 1100 T1 .EQ 4,5
06- 1110 T2 .EQ 6,7
1120 *-----
FD8E- 1130 CROUT .EQ $FD8E
FD8E- 1140 PRBYTE .EQ $FD8E
FD8E- 1150 COUT .EQ $FD8E
1160 *-----
1170 .OP 65802

1190 *-----
1200 TEST
1210 CLC ENTER NATIVE MODE
1220 XCE
1230 .DO VERSION=HACKNEY
1240 JSR BUILD.HACKNEY.TABLE
1250 .FIN

```

```

000802- C2 20      1260      REP #$20      16-BIT A-REGISTER
000804- A9 FF 03    1270      LDA ##$3FF    LARGEST VALUE TO TEST
000807- 85 00      1280      STA DIVIDEND
000809- A5 00      1290      LDA DIVIDEND
1300      .1
00080B- 20 26 08    1310      .DO VERSION=RBSC65802
00080E- 85 02      1320      JSR DIVIDE.BY.SEVEN.65802
1330      STA QUO.REM    QUO IN 15...8, REM IN 7...0
1340      .FIN
1350      .DO VERSION=HACKNEY
1360      JSR HACKNEY.DIV7
1370      STA QUO.REM    QUO IN 15...8, REM IN 7...0
1380      .FIN
1390      .DO VERSION=RBSC6502
1400      JSR DIVIDE.BY.SEVEN.6502
1410      .FIN
1420      .DO VERSION=TWO.C
1430      JSR DIV7.TWOC
1440      .FIN
000810- 20 BD 08    1450      JSR CHECK      TEST RESULT BY MULTIPLYING
000813- 90 03      1460      BCC .2        ...CORRECT ANSWER
000815- 20 92 08    1470      JSR PRINT      ...INCORRECT DIVISION
000818- 20 E8 08    1480      JSR PAUSE      CHECK FOR KEYPRESS
00081B- F0 06      1490      BEQ .3        <RET>, ABORT
00081D- C2 20      1500      REP #$20      16-BIT A-REGISTER
00081F- C6 00      1510      DEC DIVIDEND
000821- 10 E6      1520      BPL .1        ...NEXT ONE
000823- 38        1530      SEC          RETURN TO EMULATION MODE
000824- FB        1540      XCE
000825- 60        1550      RTS
1560      *-----*
1570      * QUO = VAL * .001001001001001
1580      *-----*
000826- 85 04      1590      DIVIDE.BY.SEVEN.65802
000828- 0A        1600      STA T1        SAVE ORIGINAL VALUE
000829- 0A        1610      ASL          MULTIPLY BY 64
00082A- 0A        1620      ASL
00082B- 0A        1630      ASL
00082C- 0A        1640      ASL
00082D- 0A        1650      ASL
00082E- 65 04      1660      ADC T1        ADD, EQUIV. TO * .01000001
000830- 85 04      1670      STA T1        SAVE RESULT
000832- 4A        1680      LSR          DIVIDE BY 8, WHICH IS
000833- 4A        1690      LSR          EQUIV. TO * .00001000001
000834- 4A        1700      LSR
000835- 65 04      1710      ADC T1        EQUIV TO * .01001001001
1720      .DO 0
1730      STA T1        EXTENDED PRECISION METHOD
1740      XBA          GET EQUIV. TO * .000000000000001
1750      AND ##$00FF
1760      LSR
1770      LSR
1780      LSR
1790      LSR
1800      ADC T1        EQUIV. TO * .01001001001001
1810      .ELSE
1820      CMP ##$8800    FUDGE FACTOR METHOD
1830      ADC ##$0008    ADD $0008 TO ALL VALUES,
1840      CMP ##$8800    AND $0002 MORE TO BIG ONES
1850      ADC ##$0000
1860      .FIN
000843- 4A        1870      LSR          DIVIDE BY 2, RESULT IS QUOTIENT
000844- E2 20      1880      SEP #$20      IN HI BYTE, REM IN NEXT 3 BITS
000846- 4A        1890      LSR          ISOLATE REMAINDER IN LO BYTE
000847- 4A        1900      LSR
000848- 4A        1910      LSR
000849- 4A        1920      LSR
00084A- 4A        1930      LSR
00084B- C2 20      1940      REP #$20
00084D- 60        1950      RTS
1960      *-----*
00084E- 08        1970      DIVIDE.BY.SEVEN.6502
00084F- 38        1980      PHP          SAVE M&X BITS
000850- FB        1990      SEC          SWITCH TO EMULATION MODE
000851- 08        2000      XCE
2010      PHP
2020      *-----*

```

[illegible]

One look at the chart will give you some of the reasons there's only one smart choice in 80 column cards for your Apple. But the real secret to Viewmaster 80's success is something even better: Total compatibility.

And the Viewmaster 80 delivers a super sharp, state-of-the-art display with a 7x9 character matrix for clear, easily readable characters. Here are just a few of the powerful features the Viewmaster 80 delivers for a great price (\$139):

- Call to order today; 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5½% sales tax. Add \$10.00 outside U.S.A.

Page 28.....Apple Assembly Line.....May, 1986.....Copyright (C) S-C SOFTWARE

```

000852- A5 00      2030      LDA DIVIDEND
000854- 4A         2040      LSR
000855- 4A         2050      LSR
000856- 4A         2060      LSR
000857- 65 00      2070      ADC DIVIDEND
000859- 6A         2080      ROR
00085A- 4A         2090      LSR
00085B- 4A         2100      LSR
00085C- 65 00      2110      ADC DIVIDEND
00085E- 6A         2120      ROR
00085F- 29 FC      2130      AND #$FC
000861- 85 04      2140      STA T1
000863- 4A         2150      LSR
000864- 85 06      2160      STA T2
000866- 4A         2170      LSR
000867- 85 03      2180      STA QUO.REM+1      QUO = LO-BYTE/7
000869- 65 04      2190      ADC T1
00086B- 65 06      2200      ADC T2      QUO*7
00086D- 49 FF      2210      EOR #$FF      -QUO*7
00086F- 38         2220      SEC
000870- 65 00      2230      ADC DIVIDEND      REM
000872- A6 01      2240      LDX DIVIDEND+1      0,1, OR 2
000874- 7D 8E 08   2250      ADC RTBL,X
000877- C9 07      2260      CMP #7
000879- 90 02      2270      BCC .1
00087B- E9 07      2280      SBC #7
00087D- 85 02      2290      STA QUO.REM      FINAL REMAINDER
00087F- BD 8A 08   2300      LDA QTBL,X
000882- 65 03      2310      ADC QUO.REM+1
000884- 85 03      2320      STA QUO.REM+1      FINAL QUOTIENT
000886- 28         2330      *-----*
000887- FB         2340      PLP      SWITCH TO ORIGINAL MODE
000888- 28         2350      XCE
000889- 60         2360      PLP      X&M BITS
000889- 60         2370      RTS
00088A- 00 24 49 6D 2380      *-----*
00088E- FF 03 00 04 2390      QTBL .DA #0,#36,#73,#109
00088E- FF 03 00 04 2400      RTBL .DA #-1,#3,#0,#4
00088E- FF 03 00 04 2410      *-----*
000892- 08         2420      PRINT
000893- 38         2430      PHP      SAVE M&X BITS
000894- FB         2440      SEC      SWITCH TO EMULATION MODE
000895- FB         2450      XCE
000896- 08         2460      PHP      SAVE ORIGINAL MODE (C-BIT)
000898- A5 01      2470      LDA DIVIDEND+1
000898- 09 B0      2480      ORA #0      PRINT DIVIDEND IN HEX
00089A- 20 ED FD   2490      JSR COUT
00089D- A5 00      2500      LDA DIVIDEND
00089F- 20 DA FD   2510      JSR PRBYTE
0008A2- A9 A0      2520      LDA #" "      PRINT QUOTIENT IN HEX
0008A4- 20 ED FD   2530      JSR COUT
0008A7- A5 03      2540      LDA QUO.REM+1
0008A9- 20 DA FD   2550      JSR PRBYTE
0008AC- A9 A0      2560      LDA #" "      PRINT REMAINDER IN HEX
0008AE- 20 ED FD   2570      JSR COUT
0008B1- A5 02      2580      LDA QUO.REM
0008B3- 20 DA FD   2590      JSR PRBYTE
0008B6- 20 8E FD   2600      JSR CROUT      <RETURN>
0008B9- 28         2610      PLP      RESTORE NATIVE/EMULATION BIT
0008BA- FB         2620      XCE
0008BB- 28         2630      PLP      RESTORE M&X BITS
0008BC- 60         2640      RTS
0008BC- 60         2650      *-----*
0008BD- A5 02      2660      CHECK
0008BF- 29 00 FF   2670      LDA QUO.REM
0008C2- 4A         2680      AND #$FF00      ISOLATE QUOTIENT
0008C3- 4A         2690      LSR      DIVIDE BY 64 FOR NOW
0008C4- 4A         2700      LSR
0008C5- 4A         2710      LSR
0008C6- 4A         2720      LSR
0008C7- 4A         2730      LSR
0008C8- 85 04      2740      LSR
0008C8- 85 04      2750      STA T1
0008CA- 4A         2760      LSR      MULTIPLY BY SEVEN
0008CB- 85 06      2770      STA T2
0008CD- 4A         2780      LSR
0008CE- 65 04      2790      ADC T1
0008D0- 65 06      2800      ADC T2

```

```

0008D2- 85 04      2810      STA T1          QUO * 7
0008D4- A5 02      2820      LDA QUO.REM      CHECK FOR VALID REMAINDER
0008D6- 29 FF 00   2830      AND ##$00FF      0...7
0008D9- C9 07 00   2840      CMP ##7
0008DC- B0 08      2850      BCS .1          ...INVALID REMAINDER
0008DE- 65 04      2860      ADC T1          ADD QUO*7
0008E0- C5 00      2870      CMP DIVIDEND      ...BETTER BE SAME!
0008E2- D0 02      2880      BNE .1          ...NOT, INVALID QUO & REM
0008E4- 18         2890      CLC          SIGNAL VALID ANSWERS
0008E5- 60         2900      RTS
0008E6- 38         2910      .1      SEC          SIGNAL INVALID ANSWERS
0008E7- 60         2920      RTS
2930      *-----
2940      PAUSE
0008E8- E2 20      2950      SEP ##20      8-BIT A-REGISTER
0008EA- AD 00 CO   2960      LDA $C000      CHECK KEYBOARD
0008ED- 10 0F      2970      BPL .2          NOTHING TYPED
0008EF- 8D 10 CO   2980      STA $C010      CLEAR STROBE
0008F2- C9 8D      2990      CMP ##8D      <RETURN>?
0008F4- F0 08      3000      BEQ .2          <RET>, SO DON'T PAUSE
0008F6- AD 00 CO   3010      .1      LDA $C000      SOME OTHER KEY, SO PAUSE
0008F9- 10 FB      3020      BPL .1          ...TILL ANOTHER KEY TYPED
0008FB- 8D 10 CO   3030      STA $C010      CLEAR STROBE
0008FE- C9 8D      3040      .2      CMP ##8D      .EQ. IF <RET>
000900- 60         3050      RTS          ...ELSE .NE.
3060      *-----
3070      *      DIVIDE BY 7 FROM NEW //C ROMS (AT $CB4F-CBB0)
3080      *      USED TO GET NUMBER OF 7-BYTES PACKETS
3090      *      IN A BUFFER, FOR THE PROTOCOL CONVERTER
3100      *-----
3110      DIV7.TWOC
000901- 08         3120      PHP          SAVE X&M BITS
000902- 38         3130      SEC          ENTER EMULATION MODE
000903- FB         3140      XCE
000904- 08         3150      PHP          SAVE PREVIOUS MODE
3160      *---ALGORITHM FROM //C-----
000905- A6 01      3170      LDX DIVIDEND+1      HI BYTE (0, 1, OR 2)
000907- BD 04 09   3180      LDA PDIV7TAB,X      0, $100, OR $200 DIVIDED BY 7
00090A- 85 03      3190      STA QUO.REM+1      QUOTIENT SO FAR
00090C- BD 43 09   3200      LDA PMOD7TAB,X      0, $100, OR $200 MOD 7
00090F- 85 02      3210      STA QUO.REM      REMAINDER SO FAR
3220      *---PROCESS NEXT 5 BITS-----
000911- A2 05      3230      LDX #5
000913- A5 00      3240      LDA DIVIDEND      LOW BYTE
000915- 85 04      3250      STA T1          WORKING COPY
000917- 29 07      3260      AND #7          LOW 3 BITS
000919- A8         3270      TAY          SAVE FOR LATER USE
00091A- 06 04      3280      .1      ASL T1          GET NEXT BIT FROM DIVIDEND IN CARRY
00091C- 90 15      3290      BCC .4          IF CLEAR, NO EFFECT ON QUO,MOD
00091E- BD 46 09   3300      LDA MOD7TAB,X      GET MOD7 FOR 2^N
000921- 18         3310      .2      CLC          UPDATE MOD VALUE
000922- 65 02      3320      ADC QUO.REM
000924- C9 07      3330      CMP #7          OVERFLOW?
000926- 90 02      3340      BCC .3          ...NO
000928- E9 07      3350      SBC #7          ...YES, CORRECT
00092A- 85 02      3360      .3      STA QUO.REM      REMAINDER SO FAR
00092C- BD 4C 09   3370      LDA DIV7TAB,X      GET QUOTIENT FOR 2^N
00092F- 65 03      3380      ADC QUO.REM+1
000931- 85 03      3390      STA QUO.REM+1      QUOTIENT SO FAR
000933- CA         3400      .4      DEX          ONE LESS BIT TO DEAL WITH
000934- 30 06      3410      BMI .5          ...FINISHED
000936- D0 E2      3420      BNE .1          ...FIVE TIMES
000938- 98         3430      TYA          GET BACK FIRST 3 BITS
000939- 4C 21 09   3440      JMP .2          ADD IN REMAINDER
3450      *---RETURN TO CALLER-----
00093C- 28         3460      .5      PLP          ORIGINAL MODE
00093D- FB         3470      XCE
00093E- 28         3480      PLP          RESTORE X&M BITS
00093F- 60         3490      RTS
3500      *-----
000940- 00 24 49   3510      PDIV7TAB .DA #0,#36,#73
000943- 00 04 01   3520      PMOD7TAB .DA #0,#4,#1
000946- 00 01 02 04
00094A- 01 02      3530      MOD7TAB .DA #0,#1,#2,#4,#1,#2
00094C- 00 01 02 04
000950- 09 12      3540      DIV7TAB .DA #0,#1,#2,#4,#9,#18
3550      *-----

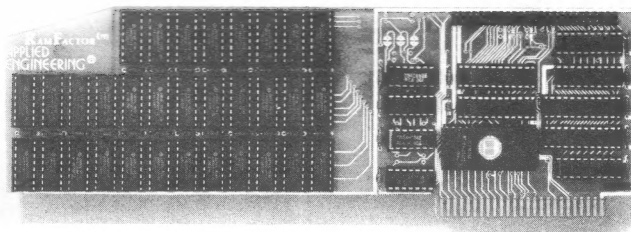
```

RamFactor™

All the Performance, Speed, and Software Compatibility of RamWorks™ in a Slot 1 through 7 Card.

That's right! Now Applied Engineering offers you a choice. While RamWorks is the clear winner for the auxiliary slot in a IIe, RamFactor is the standard for slots 1 through 7. Now anyone with an Apple II+, Franklin, or Apple IIe preferring to use slots 1 through 7 can now enjoy the speed and performance that until now was only available with RamWorks.

With RamFactor, you'll be able to instantly add another 256K, 512K, or a full 1 meg on the main board and up to 16 meg with additional piggyback card. And since virtually all software is automatically compatible with RamFactor, you'll immediately be able to load programs into RamFactor for instantaneous access to information. You'll also be able to store more data for longer word processing documents, bigger data bases, and expanded spreadsheets.



Very Compatible

All the leading software is already compatible with RamFactor. Programs like AppleWorks, Pinpoint, BPI, Managing Your Money, Dollars and Sense, SuperCalc 3A, PFS, Mouse Write, MouseDesk, MouseCalc, Sensible Speller, Applewriter IIe, Business Works, ReportWorks, Catalyst 3.0 and more. And RamFactor is fully ProDos, DOS 3.3, Pascal 3.3 and CP/M compatible. In fact, no other memory card (RamWorks excepted) is more compatible with commercial software.

AppleWorks Power

There are other slot 1-7 cards that give AppleWorks a larger desktop, but that's the end of their story. But RamFactor is the only slot 1-7 card that increases AppleWorks internal memory limits, increasing the maximum number of lines permitted in the word processor, and RamFactor is the only standard slot card that will automatically load AppleWorks into RAM dramatically increasing speed and eliminating the time required to access the program disk, it will even display the time and date on the AppleWorks screen with any ProDos clock. RamFactor will automatically segment large files so they can be saved on 5 1/4", 3 1/2", and hard disks. All this performance is available to anyone with an Apple IIe or II+ with an 80 column card.

RamFactor, no other standard slot card comes close to enhancing AppleWorks so much.

True 65C816 16 Bit Power

RamFactor has a built-in 65C816 CPU port for direct connection to our IIe 65C816 card for linearly addressing up to 16 meg for the most powerful 16 bit applications. (II+ 65C816 card under development.)

Powerful Program Switcher

With RamFactor, you can organize memory into multiple work areas and switch between them. Each work area can contain different programs and even different operating systems. Now you can switch from one program to another or even switch from AppleWorks to DOS 3.3 to CP/M to Pascal to ProDos in under a second. And with our Battery back-up option, you can have permanent storage for up to 20 years.

Quality and Support of the Industry Leader

RamFactor is from Applied Engineering, the largest, most well supported manufacturer of Apple peripherals and the inventor of large RAM cards for the Apple. With our 5 year no hassle warranty and outstanding technical support, you're assured of the most trouble free product you can buy.

Features:

- Up to 16 meg total memory, 256K to 1 meg on main board. Up to 16 meg with additional memory on piggyback card.
- Fully Apple II Memory Expansion compatible
- Compatible with Apple IIe, II+ and Franklin
- Battery back-up option allows you to turn on your Apple and run your favorite programs in less than 1 second!
- Automatically recognized by ProDos, DOS 3.3, Pascal and CP/M
- Built-in RamDrive™ software (a true RAM disk not disk caching)
- Systems are directly bootable from RamFactor if desired
- Built-in linear addressing 16 bit co-processor port
- Built-in self diagnostic software
- Automatic expansion with AppleWorks 1.3 or later
- Allows Apple II+ and IIe to run your AppleWorks without buying additional software
- Accelerates AppleWorks
- Displays time and date on the AppleWorks screen with any ProDos clock
- Fits any I/O slot except slot 3
- Fully socketed and user upgradeable
- Much, much more

RamFactor with 256K	\$239
RamFactor with 512K	\$289
RamFactor with 1 MEG	\$389
RamFactor with 2-16 MEG	CALL
Battery Back-up Option	\$179
65C816 16 Bit Card	\$159

Order RamFactor today... with 15 day money back guarantee and our "no hassle" five year warranty. Call 9 a.m. to 11 p.m., 7 days, or send check or money order to Applied Engineering. MasterCard, Visa and C.O.D. welcome. Texas residents add 5% sales tax. Add \$10.00 if outside U.S.A.

AE Applied Engineering
The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

```

000952- 85 04      3560 HACKNEY.DIV7
000954- 29 07 00   3570      STA T1      SAVE VALUE
000957- 85 06      3580      AND ##$0007  SAVE LOWER 3 BITS (MOD 8)
000959- A5 04      3590      STA T2
00095B- 4A        3600      LDA T1      DIVIDE BY 8
00095C- 4A        3610      LSR
00095D- 4A        3620      LSR
00095E- 0A        3630      LSR
00095F- AA        3640      ASL      DOUBLE FOR TABLE INDEX
000960- BD 00 0A   3650      TAX      GET QUO & REM FROM TABLE
000963- 0A        3660      LDA TABLE,X
000964- 0A        3670      ASL      MULTIPLY BOTH BY 8
000965- 0A        3680      ASL
000966- 65 06     3690      ASL
000968- AA        3700      ADC T2      ADD LOWER BITS BACK
000969- 29 00 FF   3710      TAX      SAVE RESULT
00096C- 85 04     3720      AND ##$FF00  KEEP QUOTIENT
00096E- 8A        3730      STA T1
00096F- 0A        3740      TXA      GET REMAINDER
000970- AA        3750      ASL      DOUBLE FOR INDEX
000971- BD 00 0A   3760      TAX
000974- 18        3770      LDA TABLE,X  GET QUO & REM FROM TABLE
000975- 65 04     3780      CLC
000977- 60        3790      ADC T1      ADD PREVIOUS QUOTIENT
000977- 60        3800      RTS
000978- 08        3810      #-----
000979- C2 20     3820 BUILD.HACKNEY.TABLE
00097B- A9 00 0A   3830      PHP      SAVE M&X BITS
00097E- 85 04     3840      REP ##$20  LONG A-REG
000980- E2 30     3850      LDA ##TABLE
000982- A2 00     3860      STA T1
000984- 9B        3870      SEP ##$30  ALL REGS SHORT
000985- 8A        3880      LDX #0      X = REMAINDER
000986- 92 04     3890      TXY      Y = QUOTIENT
000988- E6 04     3900      TXA      STORE CURRENT REMAINDER
00098A- 98        3910      STA (T1)
00098B- 92 04     3920      INC T1
00098D- E6 04     3930      TYA      STORE CURRENT QUOTIENT
00098F- E8        3940      STA (T1)
000990- E0 07     3950      INC T1
000992- 90 F1     3960      INX      NEXT REMAINDER
000994- A2 00     3970      CPX #7
000996- C8        3980      BCC .1      ...NO CHANGE TO QUOTIENT
000997- C0 0A     3990      LDX #0      NEXT QUOTIENT
000999- 90 EA     4000      INY
00099B- 28        4010      .DO 1
00099C- 60        4020      CPY #10      STOP AFTER QUO=9, REM=6
00099D- 4100     4030      .ELSE
00099E- 4110     4040      CPY #16      STOP AFTER QUO=15, REM=6
00099F- 4120     4050      .FIN
0009A0- 4060      BCC .1      ...NOT YET
0009A1- 4070      PLP      RESTORE M&X BITS
0009A2- 4080      RTS
0009A3- 4090      #-----
0009A4- 4100      .BS #+255/256#256-#
0009A5- 4110      TABLE .EQ #
0009A6- 4120      #-----

```

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is \$18 per year in the USA, sent Bulk Mail; add \$3 for First Class postage in USA, Canada, and Mexico; add \$14 postage for other countries. Back issues are available for \$1.80 each (other countries add \$1 per back issue for postage). A subscription to the newsletter and the Monthly Disk containing all source code is \$45 per year in the US, Canada and Mexico, and \$87 to other countries.

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)